AD-A068 267 PURDUE UNIV LAFAYETTE IND PURDUE LAB FOR APPLIED IND--ETC F/G 9/2

MINUTES 1978 SPRING REGIONAL MEETING INTERNATIONAL PURDUE WORKS--ETC(U)

JUN 78 N00014-78-C-0127

UNCLASSIFIED NL

1 OF 4

AD A068267


4
267

LEVEL II

(12)

NB 049-388

AD A068267

DDC FILE COPY

[redacted]

79 04 05 01

~~79 01 23 506~~

This Report is Published as Part of
Engineering Experiment Station Bulletin 143 Series

Schools of Engineering
Purdue University
West Lafayette, Indiana 47907

15 N00014-78-C-0127

11 30 Jun 78

12 371p.

12

6 MINUTES

1978 SPRING REGIONAL MEETING

INTERNATIONAL PURDUE WORKSHOPS

ON INDUSTRIAL COMPUTER SYSTEMS.

PART II.

TECHNICAL APPENDICES.

Contract N00014-78-C-0127

held at

FEDERAL INSTITUTE OF TECHNOLOGY (ETH)

ZURICH, SWITZERLAND

April 4 - 7, 1978,

PURDUE UNIVERSITY, WEST LAFAYETTE, INDIANA

April 10 - 12, 1978

and

JEIDA, TOKYO, JAPAN

June 29 - 30, 1978.

Purdue Laboratory for Applied Industrial Control
Schools of Engineering
Purdue University
West Lafayette, Indiana 47907

408 244 Gm

The International Purdue Workshop on
Industrial Computer Systems
is
Jointly sponsored by the
Purdue Laboratory for Applied Industrial Control;
by the Automatic Control Systems Division,
the Chemical and Petroleum Industries Division,
and the Data Handling and Computations Division
of the
Instrument Society of America;

by the
Japan Electronic Industry Development Association
(JEIDA) through the IPW Japan Committee;
and by the Commission of the European Communities
through its General Directorate for Internal Markets
and Industrial Affairs.

It is also sponsored by the
International Federation for Information Processing
as Working Group 5.4 of Technical Committee TC-5,
and by the
Associate Committee for Automatic Control,
National Research Council of Canada.

The work of the International Purdue Workshop
is
partially funded by a grant from the
Naval Air Systems Command, U.S. Navy,
through the Office of Naval Research,
Washington, D.C.

The Workshop is affiliated with the
Institute of Electrical and Electronic Engineers
through the
Data Acquisition and Control Committee
of the Computer Society
and the
Industrial Control Committee
of the Industrial Applications Society.

It is also affiliated with the
International Federation of Automatic Control
through its
Computers Committee.

TABLE OF CONTENTS

TABLE OF CONTENTS (Cont.)

TABLE OF CONTENTS (Cont.)

APPENDIX J-II

TC-3

LONG TERM PROCEDURAL LANGUAGES COMMITTEE

PURDUE JAPAN

1. Translation of JEIDA Report 52-A-117 on Industrial Computer Langauges

JEIDA Report 52-A-117 on Industrial Computer Languages

(1977 March)

Note for Reading

The report is the collection of answers to the questionnaire and therefore lacks consistency in their description. Also please note that this is the literal translation of the report.

The report consists of 19 sections; each section describes one language. Almost languages listed here are based on JIS (Japanese Industrial Standard) FORTRAN. There are three JIS FORTRAN: level 7000, level 5000 and level 3000. JIS FORTRAN level 7000 is equivalent to ANS X3.9-66 FORTRAN level 3000 to ANS X3.10-66 Basic FORTRAN. JIS FORTRAN Level 5000 is equivalent to ISO Intermediate FORTRAN.

Mr. Ed. Tokunaga, IBM Japan, Ltd., kindly provided this translation.

1. Oko Electric Co.

1. Language Name

   FORTRAN

2. Hardware

   OKITAC-50 (Model 20 and 40)

3. Base Language, Level and Design Objective

   Base Language : FORTRAN

   Level : JIS FORTRAN (Level 7000)

   Design objective: -Speeding up of processing time with using minicomputer

   -Various I/O attachment capability

4. Character Set, Label, Variable Name

   Character set : 36 alphanumerals, Katakana and special characters

   Label : Unsigned integer number (1∿32767)

   Variable name : -1∿6 alphanumerals, where the first must be alphabet

   -1∿6 Katakana characters

5. Data Types and Precision

Integer	: Single (16 bits)	$(-2^{15})\sim(2^{15}-1)$	
	Double (32 bits)	$(-2^{31})\sim(2^{31}-1)$	
Real	: Single (32 bits)	$16^{-65}\leq\|x\|<16^{63}$, 7 significant digits	
	Double (64 bits)	$16^{-65}\leq\|x\|<16^{63}$, 14 significant digits	

Logical : Available (.TRUE., .FALSE.)

Character : Internal code (ISO code), Up to 255 characters

Hexadecimal: Availabe (0∿9, A-F)

Complex

6. Default Rules

Type declaration: Implicit type declaration (I∿N rule)

Omission of incrementation parameter of DO statement is interpreted as a value of one.

7. Storage Allocation

Unknown

8. Common Variables

- Labeled common block, unlabeled common block
- External common block

9. Variables on External Memory

External common block (ARRARY Statement)

10. Array Dimension

3 dimensions

11. Structure

none

12. Built-in Functions

See attached Table 1.

13. Language Expansion Capability or Conditional Compile/Macro Facility

None

14. Character/Bit String Manipulation Capability

Available through character assignment statement, hexadecimal assignment statement, arithmetic assignment statement and IF statement.

15. Interruption Handling

- Available on some hardwares (e.g. P I/O, GD, CRT)
- Occurrence of interruption is detected by hardware and supervisor passes information to the interruption handling routine in a subroutine.

16. Input/Output

- Sequential I/O statements: READ/WRITE statements, auxiliary I/O statements

- Direct Access I/O statements: DEFINE FILE statement, direct access READ/WRITE statements FIND statement

17. File Organization

Sequential file

Direct file

18. Method of Input/Output from/to I/O Unit (TW, CRT, etc.)

1) Availability of READ/WRITE Statements

Available: - For CRT (character display), input/output is done by READ/WRITE statement as those of typewriter console.

- For Graphic Display and Colour Dot Display, input/output is performed by a subroutine (using CALL statement)

2) Others

X-Y Plotter: - Control linkage - CALL statement

- Data linkage - Arguments

- Format linkage - FORMAT statement (H type) or dummy arguments

19. Debug Facility

1) Debug Line: The character D in column 1 of a line designates the line as a debug line. If DL option is specified by programmer, the character D in column 1 is replaced by a space, and if DL option is not specified, the character D is replaced by the character C and the line is regarded as a comment line

2) Debug Facility Statements

DEBUG

AT

2) TRACE ON

TRACE OFF

DISPLAY - This statement specifies to display the name and the value of a variable and an array in the list.

20. Real Time Facilities

See attached Table 2.

21. Process I/O Facilities

See attached Table 3.

22. Other Facilities

This FORTRAN incorporates JIS FORTRAN (Level 7000). The features expanded from JIS FORTRAN (Level 7000) are listed in the attached Table 4.

23. Applicable Area

Scientific application, process control, data logging, instrumentation control, environmental testing.

2. Fujitsu/PANAFACOM

1. Language Name

FORTRAN

2. Hardware

PFU Series (PFU-100, 200, 300, 400)

3. Base Language, Level and Design Objective

Base language	:	FORTRAN
Level	:	JIS FORTRAN (Level 7000)
Design objective:		Produce high performance object programs in scientific and industrial applications

4. Character Set, Label, Variable Name

Character Set: 10 numerals, 26 alphabets, 12 special characters

TABLE 1. Built-in Functions

Name	Function
ABS	Absolute value (IABS, JABS, ABS, DABS)
INT	Trancation (INT, JINT, IDINT, JDINT, AINT, DINT
MOD	Remaindering (MOD, JMOD, AMOD, DMOD)
MAX	Largest value (MAXO, JMAXO, MAXI, JMAXI, AMAXO, AJMAXO, AMAXI, DMAXI)
MIN	Smallest value (MINO, JMINO, MINI, JMINI, AMINO, AJMINO, AMINI, DMINI)
FLOAT	Float (FLOAT, FLOATJ, DFLOAT, DFLOTJ)
FIX	Fix (IFIX, JFIX)
SIGN	Transfer of sign (ISIGN, JSIGN, SIGN, DSIGN)
DIM	Postive difference (IDIM, JDIM, DIM, DDIM)
SNGL	Conversion to single (SNGL)
REAL	Obtain real part (REAL)
AIMAG	Obtain imaginary part (AIMAG)
DBLE	Conversion to double (DBLE)
CMPLX	Conversion to complex (CMPLX)
CONJG	Conjugate of a complex argument (CONJG)

TABLE 2. Real Time Facilities

Following routines are referenced by CALL statement:

Name	Function
Activation/Termination	
ACTIV	Initiate a specified subtask
BLOCK	Make a specified task to suspended status
WAKEUP	Make a suspended task to ready or wait status
EOTASK	Inform of normal end of a task
ABEND	Inform of abnormal end of a task
Synchronization	
WAIT	Waite an event occurrence
WAITM	Wait multiple events occurrence
POST	Inform of an event occurrence
QON	Inform of an event occurrence as a queue
QOFF	Waite an event occurrence in a queue
Resource Sharing	
ENQ	Enqueue for resources
DEQ	Dequeue for resources
RESERV	Reserve resources among partitions
RELEAS	Release resources among partitions
Others	
CHAP	Change priority of a task
GETIME	Get current date and time
SETIME	Set timer
BLDPOL	Reserve a working area
GETSEL	Get cell
FRESEL	Free cell
PDUMP	Dump memory and register
DUMP	Dump task areas and register in case of normal end of a task

TABLE 3. Process I/O Facilities

Following routines are referenced by CALL Statement:

Name	Function
Definition	
PDTRAP	Define interruption sources table
PDDVA	Define DVA information of units
PDWORK	Define work area commonly used to subroutines
PDFILE	Define external file name
PDCOM	Define area commonly used to routines
PDTIME	Define information for day-time clock
PDAI	Describe analog input points information
PDDI	Describe digital input information
PDAO	Describe analog output points information
PDDO	Describe digital output information
Execution	
PAI	Analog input
PAO	Analog output
PDI	Digital input with process interrupt
PDIN	Digital input without process interrupt
PDO	Digital output with process interrupt
PDON	Digital output without process interrupt
PIN	Process of interrupt input
PRTIME	Process of real time clock
PIPLS	Pulse input
POPLS	Pulse output
PDCONT	Device control
Others	
PTEST	Sense of a speciric interruption
PWAIT	Waiting of a specific interruption

TABLE 4

FEATURES EXPANDED FROM JIS FORTRAN (LEVEL 7000)

- & '
- Hexadecimal number
- Length specification for integer and real types
- Literal enclosed in apostrophes
- Katakana symbolic name
- Mixed-mode expressions
- Mixed-mode arithmetic assignment statement
- Character assignment statement
- Hexadecimal assignment statement
- PAUSE 'message'
- Decimal numbers in PAUSE n and STOP n
- ERR and END parameters in a READ
- Direct access I/O statements
- T and Z format codes, '---', expanded G format code
- Hexadecimal number in DATA statement
- ARRAY statement
- IMPLICIT statement
- Initial data values in explicit specification statements (except DOUBLE PRECISION statement)
- Built-in functions for double precision integer value
- Dummy argument * in SUBROUTINE statement
- Statement number following & in CALL statement
- RETURN i statement
- ENTRY statement
- Debug line
- Debug facility statements

Label : Unsigned integer number (1~32767)

Variable name: 1~6 alphanumerals, where the first must be alphabet

5. Data Types and Precision

Integer: Single (16 bits) -32767 ~ 32767
Double (32 bits) -2147483647 ~ 2147483647

Real :

	Precision	Magnitude
Single (32 bits)	6.3 decimal digits	$10^{-76} \sim 10^{75}$
Double (64 bits)	15.9 decimal digits	$10^{-76} \sim 10^{75}$

Complex: Single real number

Logical

Character

Hexadecimal

6. Default Rules

Type declaration: Implicit type declaration (I~N rule)

Array subscript : The first element of the array

The word "CALL" in CALL statement may be omitted.

7. Storage Allocation

Static

8. Common Variables

Labeled common block

9. Variables on External Memory

Labeled common block

10. Array Dimension

3 dimensions

11. Structure

None

12. Built-in Functions

All of intrinsic functions and basic external functions defined in JIS FORTRAN (Level 7000) are available. And additionally, DFLOAT and DTANH are provided.

13. Language Expansion Capability or Conditional Compile/Macro Facility

None

14. Character/Bit String Manipulation Capability

None

15. Interruption Handling

Available: - Detected by hardware

- Users specify to ignore the interruption or to inform the interruption

16. Input/Output

1) Formatted READ/WRITE statements

2) Unformatted READ/WRITE statements

3) Direct access READ/WRITE statements

17. File Organization

Sequential file

Direct file

18. Method of Input/Output from/to I/O Unit (TW, CRT, etc.)

1) Availability of READ/WRITE Statements

READ/WRITE statements cover the following:

Input - SYSIN, Typewriter input, Paper tape reader, Card reader, Optical mark reader

Output - SYSLIST, Typewriter output, Paper Tape punch, Line printer, Character display

Input/Output - Cassette tape, Magnetic tape, Magnetic dram/disk

2) Others

Service packages for X-Y Plotter and Graphic Display are referenced by CALL statement.

19. Debug Facility

1) Trace facility: TRACE ON statement

TRACE ON* statement

2) Debug line:

Debug lines are regarded as either comment lines or effective statements according to specification at compile time.

20. Real Time Facilities

See attached Table 1.

21. Process I/O Facilities

See attached Table 2.

22. Other Facilities

1) Service routines for handling bits and pointers

2) Overlay

- LOAD statement and DELETE statement

- Automatic overlay facility for storage shotage in execution time

3) Capability of specifying 1∿15 entries

23. Applicable Area

Scientific application, process control, production line control, machine control, system control, etc.

3. Hitachi Ltd.

1. Language Name

SPL (Software Production Language)

TABLE 1. Real Time Facilities

Following service routines are referenced by CALL statement:

Name	Function
ENTER	Initiate a specified task
START	Initiate a specified task after the specified time delay
RSTART	Repeat START
TRNON	Initiate a specified task at a specified time of day
RTRNON	Repeat TRNON
WAIT, WOUT	Suspend its own execution and be in wait state during specificed time
IWAIT, IWOUT	Suspend its own execution and be in wait state until completion of operation of I/O statement
FCAN	Cancel the previously-issued START, RSTART, TRNON or RTRNON request
FGETUP, FPUTUP	Get/Put users parameters among tasks
FENQ	Enqueues for resources (programs, data, etc.)
FDEQ	Dequeues for resources
FSWAIT, FSWOUT	Suspend its own execution and be in wait state until one of specified events occurs
FAWAIT, FAWOUT	Suspend its own execution and be in wait state until all of specified events occur
TIME	Set and obtain time of day
SMODE	Change mode (valid/invalid) of storage protection

TABLE 1. Real Time Facilities (Cont'd)

Name	Function
SLEVEL	Change priority level of a task
FERROR	Pass information to error processing routine when an error occurs
TASKSW	Suspend its own execution and control to other executable task
TSWPST	Resume the task which suspends execution by TASKSW
FPOST	Inform of completion of an event to the task which issued FSWAIT/FSWOUT or FAWAIT/FAWOUT
FQENTR	Add a specified task to queue
FCENTR	Remove a specified task from queue
FABEND	End task in abnormal end
FGSW	Get a necessary work area from buffer pool
FRSW	Release the work area to buffer pool

TABLE 2. Process I/O Facilities

Following service routines and subroutines are referrenced by CALL statement:

I. Service Routines

Name	Function
I/O Control Service (Basic)	
AISQ	Read analog inputs in the sequential order
AISQW	Same as above. But not return to the calling program until the completion of data transfer
AIRD	Read analog inputs in a specified sequence
AIRDW	Same as above. But not return to the calling program until the completion of data transfer
AO	Write analog output in a specified sequence
DI	Read digital inputs in a specified sequence
DOM	Write the momentary digital output in a specified sequence
DOL	Write the latching digital output in a specified sequence
I/O Control Service (Extension)	
AI	Read analog input with a specified mode
PLI	Read pulse input in a specified sequence
PLID	Read count down type pulse input in a specified sequence
PLO	Write pulse string output or pulse width output in a specified sequence
DMI	Initiate direct memory input in a specified sequence
DMO	Initiate direct memory output in a specified sequence
CHICU	Control input/output for channel mode ICU

TABLE 2. Process I/O Facilities (Cont'd)

Name	Function
ICUMSK	Control interruption mask for lamp status word

II. P I/O Subroutines

Device	Function	Entry Name
DI	Read 1-bit ON/OFF input	FADI 1
	Read n-bits ON/OFF input	FADI 2
	Read n x m bits and recognize the pattern	FADI 3
	Read n digits of BCD data and convert to a specified format	FADI 4
DO	Write 1-bit ON output	FADO 1
	Write 1-bit OFF output	FADO 2
	Write n-bits output	FADO 3
	Write 1-bit ON/OFF output m times	FADO 4
	Write data with BCD format by converting from code type, integer type and floating point type	FADO 5
PI	Read pulse values accumulated in 1-bit or 12-bit counter	FAPIG
AI	Convert analog input with a specified scale	FAICNV
AO	Convert analog output with a specified scale	FAOCNV
ADC	Check of Ad converter	FADCHK
WDT	Reset watch-dog timer	FWDTR
CLK	Read control timer or read/set software timer	FCLKT

2. Hardware

HIDIC 80, HIDIC 08

3. Base Language, Level and Design Objective

Base language : PL/I Subset + PASCAL

Level : ---

Design objective: Combine procedure oriented language and problem oriented language

4. Character Set, Label, Variable Name

Character set : 36 alphanumerals, 26 special characters, 48 Katakana

Identifier :
- 1∿6 alphanumerals, where the first must be alphabet
- 1∿32 alphanumerals, Katakana and special characters (#-), where the first must be either alphabet or Katakana

5. Data Types and Precision

1) Basic data type

Integer	:	Single (16 bits)	-32767 ∿ 32767
		Double (32 bits)	-2147483647 ∿ 2147483647
Real	:	Single (16 bits)	$M \times 2^{-64} \sim M \times 2^{64}$ $(0 \leq M \leq 1 - 2^{-24}$
Logical			
Bit	:	Binary and Hexadecimal	
Character			
Pointer			
FORMAT			

2) Array data type : max. 3 dimensions

3) Structure data type : max. 3 levels

4) New data type (defined by users)

6. Default Rules

Memory allocation of variables	:	Capability to specify default memory allocation of variable. When it is omitted also, LOCAL is assumed.
Type declaration	:	INTEGER type (single precision)
Actual argument	:	Follow to default specified in the corresponding dummy argument

7. Storage Allocation

Static, Based

8. Common Variables

Labeled common block	:	Internal memory common in a task (main memory)
Global	:	Internal memory common among tasks (main memory)
Bulk	:	External memory common among tasks (secondary memory)

9. Variables on External Memory

Bulk : For the reference of a variable, there is no difference whether it is on internal memory or on external memory.

10. Array Dimension

3 dimensions

11. Structure

Available

12. Built-in Functions

See attached Table 1.

13. Language Expansion Capability or Conditional Compile/Macro Facility

Conditional compile facility

Catalog facility : Editable catalog facility

Macro facility

14. Character/Bit String Manipulation Capability

- Character string manipulation : Concatenation
- Bit string manipulation : OR, exclusive OR, AND, NOT, arithemtic shift, logical shift

15. Interruption Handling

None

16. Input/Output

1) READ/WRITE statements
2) Capability to link with a subsystem called DMS (basic data management system) which provides with file manipulation macros described in the attached Table 2.
3) Capability to link with a subsystem called ADHOC (extended data management system) which provides with file manipulation macros described in the attached Table 3.

17. File Organization

basic (BAM), direct (DAM), partitioned (PBAM), sequential (SAM), list (LIST), cyclic (CYCLIC), partitioned sequential (PSAM), partitioned list (PLIST)

18. Method of Input/Output from/to I/O Unit (TW, CRT, etc.)

1) Availability of READ/WRITE Statements

   Available. Compiler makes linkage to an execution time subsystem called IOPF. The I/O operation is carried out by IOPF in parallel.

2) Others

   None

19. Debug Facility

There is a test system, TPL for the exclusive use of this language. TPL provides with the facility to set values for test, check and print variables, etc. at desired points of a program.

20. Real Time Facilities

See attached Table 4.

21. Process I/O Facilities

Able to access Process I/O by specifying file number for Process I/O as the file reference number of READ/WRITE statements.

22. Other Facilities

1) Facility as system design description language: Design level system description by specifying abstract data and procedures.

2) Structured programming facility

3) Top-down programming facility

4) Edit catalog facility

5) POL (Problem Oriented Language) definition facility

6) Standard program definition facility

7) Compile time facilities

8) Facility to check at compile-time the validity of linkage between actual argument and dummy argument

9) Calling procedure without CALL statement

10) Keyword type description of linkage between actual argument and dummy argument

23. Applicable Area

Power system, iron and steel system, chemical system, water process control system, traffic control system, production control system, plant automation system, environment system, etc.

Table 1. Built-in Functions

Name	Function	Name	Function
ALOG	Natural Logarithm	IFIX	Fix
SIN	Sine	ISFTL	Shift operation
COS	Cosine	DSFTL	"
TANH	Hyperbolic Tangent	ISLC	Shift count
ATAN	Arctangent	DSLC	"
SQRT	Square root	BSRCH	Bit search
MOD	Remaindering	IABS	Absolute value
IDMOD	"	IDABS	"
AMOD	"	ABS	"
IDIV	Quotient and remaindering	SUBST	Subdivision of bit string
IDDIV	"	EXP	Exponent
DIV	"	ISFTA	Shift operation (arithmetic)
MAXO	Largest value	DSFTA	"
IDMAX	"	REG	Get register value
MINO	Smallest value	ADDR	Get address
IDMIN	"	FADDR	Get file address
FLOAT	Float	MEMRY	Get indirect address
DFLOT	"		

Table 2. Macros of DMS

Name	Function
BLDBF	Define buffers
SAVBF	Save buffer areas
FREBF	Free buffer areas
DFINF	Define files
OPENF	Open process
CLOSF	Close process
PONTF	Positioning for start of access
CNTRF	Skip of record (block)
GETF	Sequential access macro (Synchronous)
PUTF	"
TAKEF	Direct access macro (Synchronous)
GIVEF	"
TAKPE	Direct access macro (Asynchronous)
GIVPF	"
CHKF	Synchronize asynchronous macros
GTATF	Reference file attributes
CHATF	Change file attributes

Table 3. Macros of ADHOC

Classification		Name	Function
Definition macro	Buffer definition macro	BLDBF	Define buffer pool
	Logical file definition macro	DECLF	Define a logical file
Data manipulation Macro	Common manipulation macro	OPEN	
		CLOSE	
		INTL	Initialize a file
		GCRRN	Get current record number
		GTATR	Get a file attribute
		CHATR	Change a file attribute
	File manipulation macro	CNTRL	Specify process direction in sequential
		POINT	Positioning to the first record to process
		BLDP	Build or expand partition
		DLTP	Delete partition
		FNDP	Positioning to the partition to process
		DRCP	Find or modify user area of a partition or change partition name
		REFER	Direct reference of records
		GET	Serial reference of records
		UPDAT	Update current record
		ADD	Direct addition of a record
		PUT	Serial addition of a record
		DELET	Delete current record
		QUEIN	Store a record
		QUEOT	Fetch a record
	Link manipulation macro	CNTCL	Specify process direction in sequential processing of child records
		PNTCL	Positioning to the first record in sequential processing of child records
		GETCL	Serial reference of child records
		GADCL	Serial reference of address of child records
		ADDCL	Connect parent record and child record
		DLTCL	Disconnect the specified record from the parent record
		DLTCA	Disconnect all child records from the specified parent record
		GETPL	Get the parent record of the specified child record
		GADPL	Get the address of the parent record of the specified child record

Table 4. Real Time Facilities

Following macros are provided:

Name	Function
ABORT	Terminate irregularly a specified task to be DORMANT, and release resources
RLEAS	Make a specified task in DORMANT state to be IDLE
QUEUE	Make a specified task to be RUNNABLE
SUSP	Make a specified task to be SUSPENDED
RSUM	Release the SUSPENDED state of a specified task
ASUSP	Make all other tasks than itself to be SUSPENDED
ARSUM	Release the SUSPENDED state of all SUSPENDED tasks
STOP	Terminate irregularly its own execution to make itself SCHEDULE or IDLE state, and release resources
TIMER	According to the value of parameters, - Initiate a specified task after specified time delay, - Initiate a specified task at a specified time of day, - Initiate a specified task after specified time delay and initiate it at intervals of specified time, or - Initiate a specified task at a specified time of day and initiate it at intervals of specified time
STIME	Set or reset current time of day
GTIME	Read current time of day
CTIME	Remove a timer table which a specified task uses from a queue
DELAY	Suspend its own execution and pass control to another task after setting the task of itself to be resumed after specified time delay
CHAP	Change priority of a specified task
WAIT	Suspend its own execution to wait for an event and pass control to other RUNNABLE task
POST	Make an event-waiting-task to be RUNNABLE from SUSPEND
DFECB	Enable ECB (Event Cont- l Block)
GFACT	Get factor

Table 4. Real Time Facilities (Cont'd)

Name	Function
SFACT	Set a specified factor to the table of a specified task
RESERVE	Reserve a specified area exclusively
FREE	Free the reserved area
ROPRH	Inhibit roll out temporarily in case of temporary task
ROPMT	Release ROPRH

4. Hitachi Ltd.

1. Language Name

   PCL (Process Control Language)

2. Hardware

   HIDIC 80, HIDIC 08

3. Base Language, Level and Design Objective

   Base language : FORTRAN + Some Concepts of PL/I

   Level : JIS FORTRAN (Level 7000)

   Design objective : High level language for online realtime control system

4. Character Set, Label, Variable Name

   Character set : 36 alphanumerals, 26 special characters, 48 Katakana

   Statement number : Unsigned integer number (1~9999)

   Identifier : 1 ~ 6 alphanumerals, where the first must be alphabet

5. Data Types and Precision

   Integer : Single (16 bits) -32767 ~ 32767
   Double (32 bits) -2147483647 ~ 2147483647

   Real : Single (32 bits) $M \times 2^{-64} \sim M \times 2^{64}$ $(0 \leq M \leq 1-2^{-24})$

   Logical

   Character : Internal code (ASCII code)

   Hexadecimal : Single (4 hexadecimal digits)
   Double (8 hexadecimal digits)

   Pointer (Address constant)

6. Default Rules

   Type declaration : I~N rule

   Array subscript : The first element of the array

   Lower bound value of array : 1

7. Storage Allocation

Static, Based

8. Common Variables

Labeled common block : Common in a task, internal memory (main memory)

Global : Common among tasks, internal memory (main memory)

Bulk : Common among tasks, external memory (secondary memory)

9. Variables on External Memory

Bulk : (Same as SPL of Hitachi. See item 9 of Section 3)

10. Array Dimension

3 dimensions

11. Structure

Available

12. Built-in Functions

Same as SPL of Hitachi. See item 12 of section 3.

13. Language Expansion Capability or Conditional Compile/Macro Facility

Conditional compile facility : Inclusion or deletion of debugging statements

Catalog facility

14. Character/Bit String Manipulation Capability

Bit string : OR, exclusive OR, AND, NOT, arithmetic shift, logical shift

15. Interruption Handling

None

16. Input/Output

Same as SPL of Hitachi. See item 16 of section 3.

17. File Organization

Same as SPL of Hitachi. See item 17 of section 3.

18. Method of Input/Output from/to I/O Unit (TW, CRT, etc.)

1) Availability of READ/WRITE Statements

Same as SPL of Hitachi. See item 18. (1) of section 3.

2) Others

None

19. Debug Facility

1) DEBUG VALUE statement

DEBUG FLOW statement

DEBUG SUBSCRIPT statement

2) Same as SPL of Hitachi. See item 19 of section 3.

20. Real Time Facilities

Same as SPL of Hitachi. See item 20 of section 3.

21. Process I/O Facilities

Same as SPL of Hitachi. See item 21 of section 3.

22. Other Facilities

1) Declare constant : VALUE statement among tasks
PARAMETER statement in a task

2) Compiler directing statements

3) Facility of picture conversion in FORMAT statement

4) Multiple assignment statement

5) Upper and lower dimension bound specification in array declarator

23. Applicable Area

Same as SPL of Hitachi. See item 23 of section 3.

5. Nippon Electric Co. (NEC)

1. Language Name

FORTRAN

2. Hardware

NEAC Series 3200 Model 70

3. Base Language, Level and Design Objective

Base language	:	FORTRAN
Level	:	JIS FORTRAN (Level 7000)
Design objective	:	Compiler designed to use in scientific and industrial applications and communication control

4. Character Set, Label, Variable Name

Character set	:	26 alphabets, 10 numerals, 14 special characters
Label	:	Unsigned integer number (1~99999)
Variable name	:	1 ~ 6 alphanumerals, where the first must be alphabet

5. Data Types and Precision

Integer	:	Single (16 bits)	$-2^{15} \sim 2^{15}-1$		
		Double (32 bits)	$-2^{30} \sim 2^{30}-1$		
			Precision	Magnitude	
Real	:	Single (32 bits)	23 bits	8 bits (XS-128 code)	
		Double (48 bits)	39 bits	8 bits (XS-128 code)	
Complex	:	Single real number			
Hexadecimal					
Logical					
Character	:	Internal code (JIS 8-bit code)			
Bit					
Address variable	:	Available by pointer variable subroutines			

6. Default Rules

Type declaration	:	I~N rule
Array subscript	:	The first element of the array

7. Storage Allocation

Static

8. Common Variables

Labeled common block

9. Variables on External Memory

None

10. Array Dimension

3 dimensions

11. Structure

None

12. Built-in Functions

All of intrinsic functions and basic external functions defined in FORTRAN (Level 7000) are available.

13. Language Expansion Capability or Conditional Compile/Macro Facility

1) The line whose first column is * indicates a debug line which is compiled only when a debug mode compile.

2) Mixed mode expression

3) Code conversion by ENCODE and DECODE statements

4) Bit operation by IAND, IOR, NOT, IEOR and ISHT statements

14. Character/Bit String Manipulation Capability

None

15. Interruption Handling

None

16. Input/Output

READ/WRITE statements (Not use buffer)

GET/PUT subroutines (Use buffer)

17. File Organization

Sequential file

Direct file

18. Method of Input/Output from/to I/O Unit (TW, CRT, etc.)

1) Availability of READ/WRITE Statements

READ/WRITE statements are effective to console typewriters, line printers, card readers, paper tape readers/punches, disk drives and magnetic tape units.

2) Others

Subroutines are provided for X-Y plotter, CRT, etc.

19. Debug Facility

TRACE statement : Item tracing, area tracing

20. Real Time Facilities

See attached Table 1.

21. Process I/O Facilities

1) Data Communication Facilities : See attached Table 2.

2) Process I/O Facilities : CALL statement refers to the macro named PIOC which requests I/O operation to the process I/O controller

22. Other Facilities

1) Segmentation facility

2) An abridged statement FORMAT (V) may be used in case of reading into/writing from a variable

3) Multiple entries in subroutine

4) By connection with object program generated by assembler language, an object type program can be made

23. Applicable Area

Scientific application, data communication, traffic control, environment protection control, medical control, process control, etc.

Table 1. Real Time Facilities

Macro	Function
WAIT	Make a specified task or macro-executing-task to be WAIT at a specified event
POST	Inform of a specified event occurrence to a specified task and release the wait state
RQST	Regist a specified processing request to the task queue of a specified task, and initiate the task
RTRV	Get and move to a specified area one processing request from the task queue of macro-executing-task
GTBF	Get one buffer from a specified buffer pool
PTBF	Return a specified buffer to a specified buffer pool
RENQ	Enqueue for a specified resource
RDEQ	Dequeue for a specified resource
RQTM	Make a program to be WAIT state during specified time
LOAD	Load the absolute program with a specified name and a specified number from the predefined unit
ESTK	Enable stack interrupt
DSTK	Disable stack interrupt
DATE	Get current date
CLOK	Get current time
TSKS	Get task status
ODSP	Output display information to status display equipment
IDSP	Input display information from status display equipment
SNAP	Snap shot trace
SWON	Set logical sense switch
SWOF	Reset logical sense switch
SWCA	AND-type check of logical sense switch
SWCO	OR-type check of logical sense switch
TRGR	Initiate a specified task

Table 1. Real Time Facilities (Cont'd)

Macro	Function
EXIT	Terminate macro-executing-task
INHB	Inhibit from initiating a specified task
ALOW	Allow to initiate a specified task
STIM	Initiate a specified task or Control-DSR at intervals of specified time
RTIM	Reset STIM
EOP	End of program
UEP	Unusual end of program
JOBS	Get information of next job from JCB
ASGN	Inform of assignment information on peripheral equipment for the job

Table 2. Data Communication Facilities

Following macros are referenced by CALL statement:

Name	Function
CIOC	Communication input/output control
CPUT	Communication PUT
CGET	Communication GET
DEAL	Deallocate a message from communication file
ACPT	Accept a message from CIQ
DSPL	Add a message to COQ
RERL	Reroute a message of a unusual terminal
RQUE	Return a message to a queue
RBLK	Return a block to a queue
QINL	Initialize a queue
TSTQ	Test a queue
CNCL	Cancell all blocks or the last one block of the current message
JPUT	Broadcast communication PUT
JENQ	Broadcast communication enqueue
JGET	Broadcast communication GET
JDEC	" " decrement
JDAL	" " deallocation
CNVT	Convert code of data
PTCK	Parity check
CLRC	CRC/LRC check
CRCK	CRC check
CRCC	CRC/LRC check and code conversion
CRMV	CRC/LRC check and data transfer
CRCV	CRC/LRC check, code conversion and data transfer
CRTR	Transparent data processing and move and CRC/LRC check

6. Fuji Electric Mfg. Co.

1. Language Name

FORTRAN (UMOS/D FORTRAN)

2. Hardware

PANAFACOM U Series 100/200/300/400

3. Base Language, Level and Design Objective

Base language : FORTRAN

Level : JIS FORTRAN (Level 7000)

Design objective : Effective programs in industrial and scientific applications

4. Character Set, Label, Variable Name

Character set : 26 alphabets, 10 numerals, 12 special characters

Label : Unsigned integer number (1 ~ 32767)

Variable name : 1 ~ 6 alphanumerals, where the first must be alphabet

5. Data Types and Precision

Integer : Single $-2^{15} \sim 2^{15}-1$

Double $-2^{31} \sim 2^{31}-1$

Real :

	Precision	Magnitude
Single	6.3 decimal digits	$10^{-76} \sim 10^{75}$
Double	15.9 decimal digits	$10^{-76} \sim 10^{75}$

Complex

Logical

Character : EBCDIC (8 bit)

Hexadecimal number : can be used as an actual argument

Bit

Pointer variable : can be used as an actual argument

6. Default Rules

Type declaration : I-N rule

The word "CALL" in CALL statement may be omitted

7. Storage Allocation

Static

8. Common Variables

Labeled common block, where the name is SCOM

9. Variables on External Memory

External common (bulk core memory)

10. Array Dimension

3 dimensions

11. Structure

None

12. Built-in Functions

All of intrinsic functions and basic external functions defined in JIS FORTRAN (Level 7000) are available. Extended functions are DFLOAT and DTANH.

13. Language Expansion Capability or Conditional Compile/Macro Facility

By using compile mode option:

1) The word "CALL" in CALL statement may be omitted

2) A hexadecimal constant (written as Z'XXXX') may be used as an actual argument

3) An address variable (written as A'variable name') may be used as an actual argument

4) Provide interface to users special I/O equipment

14. Character/Bit String Manipulation Capability

None

15. Interruption Handling

Users can specify as compile mode option the execution of interruption handling facility (output of error message).

16. Input/Output

READ/WRITE statements

Auxiliary I/O statements (REWIND, BACKSPACE and ENDFILE)

GET/PUT subroutines

17. File Organization

Sequential file

Direct file

Other file organizations (through subroutine call)

18. Method of Input/Output from/to I/O Unit (TW, CRT, etc.)

1) Availability of READ/WRITE Statements

Available

2) Others

- Console typewriter is accessed through FWTO and FWTOR (See item 20.)
- Logging typewriter is supported through Output Control Package.
- For CRT, display of charactors and graph and input from keyboard are supported through Character Display Subroutine Package.
- Also for CRT, the facility to produce users unique FORMAT statements is provided and is referenced by READ/WRITE statements.

19. Debug Facility

TRACE ON statement

TRACE ON * statement

TRACE OFF statement

20. Real Time Facilities

See attached Table 1.

21. Process I/O Facilities

See attached Table 2.

22. Other Facilities

- Compile time options:

1) Output of object program map

2) Output list of object program in assembler mnemonic codes

3) Object program which will overlay subroutines automatically in the case of shortage of main memory at execution time

4) Object program which posts the number of arguments to sub-programs

- Provides with following packages:

1) Output Control Routine Package (OCP)

2) Character Display Subroutine Package (CSP)

3) Data Logging Program Package (LGS)

4) Sequence Control Program Package (SECOP)

5) Communication Control Program Package (CCPP)

6) PIO Debug Aid Routine

23. Applicable Area

Process control, scientific application, language processing, graphic processing, etc.

Table 1. Real Time Facilities

Following subroutines are referenced by CALL statement or the form "name (a_1, a_2, ... a_n)" :

Name	Function
GETSA	Secure an area in a single area of temporary task area
FWTO	Print out a message on console typewriter
FWTOR	Print out a message on console typewriter and request the answer from operator
SEGLD	Request overlay type segmentation
SEGCL	Call the loaded subprogram by specifying the entry address of the subprogram

The other real time facilities provided for this language are the same as those listed in Table 1 of Section 2 (Fujitsu/PANAFACOM).

Table 2. Process I/O Facilities

Following subroutines are referenced by CALL statement or the form of "name (a_1, a_2, ... a_n)" :

Name	Function
DIN	Read n bits ON/OFF input
XCI	Read n digits of BCD data and convert to a specified data format
DON	Write n bits ON/OFF output
XCO	Convert integer or floating point data to a specified data format and display it
DOC	Write 1 bit ON/OFF controlled output
AIN AIL AINL	Read analog input of n points
AIAUT	Read analog input point after evaluating the best gain of amp for it
RPI	Read integrated value of pulse input point
RCPI	Read integrated value of pulse input point and clear integrating area and pulse counter
IPI	Clear pulse input integrating area and pulse counter
AO	Write analog output point
PO	Write pulse output
POCHK	Check output status
POCNL	Cancel output request
CLKA	Read or modify time of clock device and convert it to a specified format

7. Hokushin Electric Works, Ltd.

1. Language Name

ORTOS3 FORTRAN

2. Hardware

HOC900/15, HOC900/35

3. Base Language, Level and Design Objective

Base language	:	FORTRAN
Level	:	JIS FORTRAN (Level 7000)
Design Objective	:	FORTRAN covering all the user tasks for on line realtime system

4. Character Set, Label, Variable Name

Character set	:	36 alphanumerals, 13 special characters
Label	:	Unsigned integer number (5 digits)
Variable name	:	I ~ 6 alphanumerals, where the first must be alphabet

5. Data Types and Precision

Integer : Single (16 bits) $-32768(-2^{15}) \sim 32767(2^{15}-1)$

Real :

	Precision	Magnitude
Single (32 bits)	7 decimal digits	$10^{-39} - 10^{38}$
Double (64 bits)	16 decimal digits	$10^{-39} - 10^{38}$

Character	:	Internal code (ASCII code)
Bit	:	By means of subprogram for bit manipulation
Complex	:	Single real number
Address variable	:	By means of subprogram for address variables
Byte	:	8 bits $-128(-2^{7}) \sim 127(2^{7}-1)$

Logical

Octal number

Hexadecimal number

6. Default Rules

Type declaration : I-N rule

An array name appeared in DATA statement and I/O list specifies all of the array elements in the array.

7. Storage Allocation

Static

8. Common Variables

Labeled common block

9. Variables on External Memory

None

10. Array Dimension

3 dimensions

11. Structure

None

12. Built-in Functions

All of intrinsic functions and basic external functions defined in JIS FORTRAN (Level 7000) are available.

13. Language Expansion Capability

None

14. Character/Bit String Manipulation Capability

Bit string manipulation is available by calling system functions.

15. Interruption Handling

Interruption is detected by hardware and the monitor is executed as a task.

16. Input/Output

READ/WRITE statements

BIOC subroutine: Available both of return after the completion of I/O operation and return regardless of the completion of I/O operation.

17. <u>File Organization</u>

Random access variable length record. Record size can be dynamically changed in a program by calling OPEN subroutine.

18. <u>Method of Input/Output from/to I/O Unit (TW, CRT, etc.)</u>

1) Availability of READ/WRITE Statements

Available

2) Others

Call BIOC subroutine (See above item 16).

19. <u>Debug Facility</u>

Check of lower bound of array subscript

20. <u>Real Time Facilities</u>

See attached Table 1.

21 <u>Process I/O Facilities</u>

Process I/O such as AI, AO, DI, DO, single shot, etc. are performed by calling BIOC routine and passing parameters.

22. <u>Other Facilities</u>

1) Overlay facility : A utility program makes a load module to be overlay structure

2) Random access of a file

3) System functions are provided other than built-in functions (See attached Table 2.)

23. <u>Applicable Area</u>

Process control, instrumentational application, production line management, system control, logging, etc. /

Table 1. System Subroutines

Name	Function
ENTER	Initiate a task
ABORT	Cancel the initiation of a task
DENTER	Execute a task after specified time delay
DELY	Suspend execution of a task during specified time
WAIT	Wait for occurrence of an event
POST	Inform occurrence of an event
LOCK	Interlock resource
UNLOCK	Release the interlocked resource
SYSREP	Print out system report
GETTNO	Get the task number and level of its own
INHRO	Inhibit roll-out
ENARO	Enable roll-out
MASK	Inhibit from executing other tasks than itself
UNMASK	Release MASK
CLOCK	Get current time of day
STCLK	Set clock
BIOC	Perform input/output operation
START	Initiate a task after specified time delay
FOREP	Print out errors and execute error processing routines
ISTORE	Store value of two words to a location within core
SUSP	Select main area
VPOST	Request POST of completion of I/O operation by virtual device
DEREC	Inform error recovery of I/O device

Table 2. System Functions

Name	Function
UMUL (I, J)	I x J
UDIV (D, I)	D ÷ I giving quontient to the first word of D and remainder to the second word of D
UMLDV (D, I, J)	D x I/J
DADD (D1, D2)	D1 + D2
DSUB (D1, D2)	D1 - D2
DFLT (D)	Conversion from 2-word integer to real
DFIX (R)	Conversion from real to 2-word integer
IASR (I, N)	Arithmetic shift
ILSF (I, N)	Logical shift
DASF (D, N)	2 word length arithmetic shift
DLSF (D, N)	2 word length logical shift
DWI (I, J)	Make 2-word integer with I and J
IHIGH (D)	Get the first word of 2-word integer
ILOW (D)	Get the second word of 2-word integer
IOR (I, J)	Bit manipulation - logical OR
IAND (I, J)	" - logical AND
NOT (I)	" - logical NOT
IEOR (I, R)	" - exclusive OR
ISHFT (I, N)	Logical shift
IBSET (I, N)	Set Nth bit of I to 1
IBCLK (I, N)	Reset Nth bit of I to 0
LBTST (I, N)	Test Nth bit of I
ISCAN (I)	Scan the first bit of 1
IDRAW (I, N, M)	Shift Nth - Mth bits of I to M bits rightward
INSERT (I, N, M, J)	Insert J shifted M bits leftward into Nth - Mth bits of I
IADDR (V)	Get address of variable V
ILOAD (I)	Get 1-word data from I address
ALOAD (I)	Get 2-word data from I address

8. Japan Radio Co.

1. Language Name

FORTRAN V

2. Hardware

JAC 150/40, JAC 150/80

3. Base Language, Level and Design Objective

Base language	:	FORTRAN
Level	:	JIS FORTRAN (Level 7000)
Design objective	:	Optimizing compiler with real time facility for scientific applications

4. Character Set, Label, Variable Name

Character set	:	36 alphanumerals, 12 special characters
Label	:	Unsigned integer number (1 ~ 99999)
Variable name	:	1 ~ 6 alphanumerals, where the first must be alphabet

5. Data Types and Precision

Integer	:	Single (16 bits)	-32767 ~ 32767	
		Double (32 bits)	-2147483647 ~ 2147483647	
Real	:		Precision	Magnitude
		Single (32 bits)	6 decimal digits	10^{-79} - 10^{75}
		Double (64 bits)	16 decimal digits	10^{-79} - 10^{75}
Complex	:	Single real number		
Logical				
Character	:	Internal code (ISO/ASCII code)		
Address				
Hexadecimal number				
Bit				

6. Default Rules

Type declaration : I-N rule

7. Storage Allocation

Based

8. Common Variables

Labeled common block

9. Variables on External Memory

Global labeled common

10. Array Dimension

3 dimensions

11. Structure

None

12. Built-in Functions

All of intrinsic functions and basic external functions defined in JIS FORTRAN (Level 7000) are available. In addition to them, functions of attached Table 1 are provided.

13. Language Expansion Capability or Conditional Compile/Macro Facility

Conditional compiling

In-line assembler source

14. Character/Bit String Manipulation Capability

Available

15. Interruption Handling

Interruption is detected by hardware and OS links to interruption handling routine in a program. Every task may specify to enable or disable interruption.

16. Input/Output

READ/WRITE statements

RDR/WRTR subroutines

SYSIO subroutines (same as OS I/O function)

17. File Organization

Indexed file (Sequential, Random)

Chaining file (Sequential, Random)

Consecutive file (Sequential, Random)

18. Method of Input/Output from/to I/O Unit (TW, CRT, etc.)

1) Availability of READ/WRITE Statements

Available

2) Others

None

19. Debug Facility

1) Trace : Variables, statement numbers or unconditional trace

2) Check of upper and lower bounds of subscripts of arrays

20. Real Time Facilities

See attached Table 2.

21. Process I/O Facilities

See attached Table 3.

22. Other Facilities

1) Multiple entries in subroutine

2) List processing facility

3) Capability of defining and deleting a file during execution

4) Common sharing of subroutines among tasks

23. Applicable Area

Scientific application, process control, system control, instrumentational application, etc.

Table 1. Built-in Functions (Additional)

Name	Function
ABS	Absolute value (IABS2)
INT	Truncation (INT2, IDINT2)
MOD	Remaindering (MOD2)
MAX	Largest value (AMAX02, MAX02, MAX12)
MIN	Smallest value (AMIN02, MIN02, MIN12)
FLOAT	Float (FLOAT2)
FIX	Fix (IFIX2)
BTEST	Test of bits (BTEST)
AND	AND of bits (IAND, IAND2)
EOR	Exclusive OR of bits (IEOR, IEOR2)
OR	OR of bits (IOR, IOR2)
NOT	NOT of bits (NOT, NOT2)
SHFT	Shift of bits (ISHFT, ISHFT2)
VAL	Address reference (IVAL, IVAL2, FVAL, DVAL)
SIGN	Transfer of sign (ISIGN2)
DIM	Positive difference (IDIM2)

Table 2. Real Time Facilities

Following subroutines are referenced by CALL statement:

Name	Function
LOAD	Transfer another task on external storage to memory
START	Initiate another task
TRNON	Initiate another task at a specified time of day
ABORT	Terminate the execution of another task
EXIT	Terminate its own execution
EST	Make a task resident in memory
UNEST	Make a task non-resident in memory
HOLD	Hold execution of a task
RELSE	Release the HOLD state by HOLD subroutine
CHNGE	Change the priority of task
STTSK	Get the status of a task
QUEUE	Transfer a parameter to another task
SNDMSG	Transfer a message to another task
CFILW	Create a file
DFILW	Delete a file
OPENW	Allocate a file
CLOSE	Release the allocation of a file
MODAP	Modify access privilege of a file
RENAME	Change a file name
CON	Connect a task to trap generator
THAW	Enable trap generator
SINT	Simulate traps from trap generator
FREZE	Disable trap generator
UNCON	Disconnect a task from trap generator
SYSIO	Perform all OS I/O functions
IOERR	Write on console an error message of SYSIO error
CONMSG	Write a message on console
IFETCH	Transfer a subroutine to memory as overlay
DATE	Get the current date
TIME	Get the current time (character image)
ICLOCK	Get the current time
WAIT	Suspend its own execution during specified time
ENABLE	Enable a task trap
DISABL	Disable a task trap
INIT	Initialize a task trap

Table 3. Process I/O Facilities

Following subroutines are referenced by CALL statement:

Name	Function
AISQW	Read analog data in sequential order
DIWH	Read digital data in hand shake mode
DIW	Read digital data groups
AIRDW	Read analog data in random order
AOW	Write analog data in specified order
DOLW	Write digital data groups

9. Yamatake Honeywell

1. Language Name

   FORTRAN-700

2. Hardware

   HS716

3. Base Language, Level and Design Objective

   Base language : FORTRAN

   Level : JIS FORTRAN (Level 7000)

   Design objective : Improve flexibility for industrial application with rich facilities of process I/O and inline assembly

4. Character Set, Label, Variable Name

   Character set : 36 alphanumerals, 16 special characters

   Label : Unsigned integer number (1 ~ 32767)

   Variable name : 1 ~ 6 alphanumerals, where the first must be alphabet

5. Data Types and Precision

   Integer : Single (16 bits) -32767 ~ 32767

   Real : Single (32 bits) $10^{-38} < |x| < 10^{38}$
   Precision 6 decimal digits

   Double (48 bits) $10^{-38} < |x| < 10^{38}$
   Precision 11 decimal digits

   Hexadecimal number : Available as constant

   Logical

   Character : Internal code (ASCII-8)

   Complex

   Address constant

   Octal number : Available as constant

6. Default Rules

   Type declaration : I-N rule

7. Storage Allocation

Static, Control.

8. Common Variables

Labeled common block

Unlabeled common block

EXTERNAL/GLOBAL statement

9. Variables on External Memory

Unavailable

10. Array Dimension

No limitation so long as the size of an array is within 16 K words. (max. 2047)

11. Structure

None

12. Built-in Functions

All of intrinsic functions and basic external functions defined in JIS FORTRAN (Level 7000) are available.

13. Language Expansion Capability or Conditional Compile/Macro Facility

Conditional compile

Inline assembly facility

14. Character/Bit String Manipulation Capability

None (However, subroutines are provided for bit string manipulation)

15. Interruption Handling

None

16. Input/Output

- READ/WRITE statement (sequential file
- Subroutine call (random file) based on ISA S61.2

17. File Organization

Random access file

Sequential access file

18. Method of Input/Output from/to I/O Unit (TW, CRT, etc.)

1) Availability of READ/WRITE Statements

FOR CRT and T/W output, READ/WRITE statements are concerned with up to linkage to subsystem.

2) Others

19. Debug Facility

1) TRACE statement : a) Items trace
b) Area trace

2) SUBCHECK statement : Check the boundary of an array

20. Real Time Facilities

See attached Table 1.

21. Process I/O Facilities

See attached Table 2.

22. Other Facilities

1) IMPLICIT statement

2) DECLER/NO DECLER statements for allocation control of variables/arrays

3) An error parameter in READ/WRITE statements

4) Inline assembly: Variables, arrays, labels, etc. can be referred both by FORTRAN and inline assembly

5) Comment(*) facility

6) Trace specification at compile time

7) Mixed mode expression

8) All types of integer expressions are available as subscript of an array

9) All types of arithmetic expressions are available as parameters of DO and computed GO TO statements

23. Applicable Area

Scientific application, process control, production line control, instrumentational application, etc.

Table 1. Real Time Facilities

1) Following subroutines are referenced by CALL statement:

Name	Function
PROCED	Initiate a specified program and become wait state by itself
QUIT	Make the program which issued PROCED to resume and terminate its own execution
TRNON	Initiate a program at a specified time of day
START	Initiate a program after specified time delay
WAIT/DELAY	Become wait state during specified time
OVERLY	Load overlay codes
TIME	Read current time of day
DATE	Read current date
SYSGDT	Read current date and time as ASCII code
ISYSPA ISYSSB	Read a parameter

2) Macros for the real time monitor can be used by means of in-line assembly

Table 2. Process I/O Facilities

Following subroutines are referenced by CALL statement:

Name	Function
AISQW	Read analog input data in sequential order
AIRDW	Read analog input data in random order
AOW	Write analog output points
DIW	Read digital input points
DOMW	Write digital signal during specified time
DOLW	Write digital signal
RANGE	Convert analog input data to an industrial unit
ASET	Set address data of input/output points resulting from above subroutines to data highway
ACCSS ACCESS	Access equipments (including PIO) connected to data highway

10. Takeda Riken Mfg. Co.

1. Language Name

MOSLAB R/T FORTRAN IV

2. Hardware

NOVA and ECLIPS (Japan Mini-Comp Co.)

3. Base Language, Level and Design Objective

Base language	:	FORTRAN
Level	:	JIS FORTRAN (Level 7000)
Design objective	:	Programming by FORTRAN for sensor-based data collection, control and processing in process and instrumental industry or in laboratory

4. Character Set, Label, Variable Name

Character set	:	36 alphanumerals
Label	:	Numerals in columns 1 ~ 5 Leading zeros are meaningful (for example 12 differs from 0012)
Variable name	:	1 ~ 31 alphanumerals, where the first must be alphabet

5. Data Types and Precision

Integer : Single (16 bits) $-2^{15}-1 \sim 2^{15}-1$

Real :		Precision	Magnitude
	Single (32 bits)	6 decimal digits	$2.4\times10^{-78} \sim 7.2\times10^{75}$
	Double	16 decimal digits	$2.4\times10^{-78} \sim 7.2\times10^{75}$

Logical

Character : ASCII code

Bit : Available by means of subroutines, library functions and logical expressioı

Complex : Single (64 bits)
Double (128 bits)

6. Default Rules

Type declaration : I~N rule

Array subscript : The first element of the array

Mixed mode expression is available

7. Storage Allocation

Static, Automatic, Reentrant

8. Common Variables

- Labeled common block, blank common block
- Scalar variables and DIMENSION variables are independent among tasks

9. Variables on External Memory

EXTENDED ARRAY and extended variables are not available.

10. Array Dimension

128 dimensions

11. Structure

None

12. *Built-in Functions*

All of intrinsic functions and basic external functions defined in JIS FORTRAN (Level 7000) are available. In addition to them, provided are DATN2, DCCOS, DCSIN, SINH, TAN, DTAN, DTANH, DAIMAG, DCLOG, DCABS, DCMPLX, DCONJG, DCEXP, DFLOAT, DCSQRT, IAND, IOR, NOT, IEOR, ISHIFT, ITEST and BTEST.

13. Language Expansion Capability or Conditional Compile/Macro Facility

Conditional compile

14. Character/Bit String Manipulation Capability

None

15. Interruption Handling

None

16. Input/Output

READ/WRITE statement

RDBLK/WRBLK subroutine ..... by block (256 words) } Disk only

READR/WRITR subroutine ..... by record (optional) }

MTDIO subroutine .... I/O to MT in free format

17. File Organization

Sequential file

Random file

Consecutive file

18. Method of Input/Output from/to I/O Unit (TW, CRT, etc.)

1) Availability of READ/WRITE Statements

Available. Display to CRT is described by means of READ/WRITE statements, but graphs are handled by CALL statement.

2) Others

XY plotter is handled by CALL statement

19. Debug Facility

None

20. Real Time Facilities

See attached Table 1.

21. Process I/O Facilities

See attached Table 2.

22. Other Facilities

1) Array in negative direction: e.g. I (-4, 6)

2) Multi-file facility : up to 100 files in a reel

3) Free format I/O facility for MT

23. Applicable Area

Instrumentational application /

Table 1. Real Time Facilities

Following subroutines are referenced by CALL statement:

Name	Function
ABORT	Terminate a specified task
AKILL	Terminate all the tasks with a specified priority-number
APPEND	Open a file to append
ARDY	Suspend all the tasks with a specified priority-number
ASSOC	Assign an ID number to a task and suspend it
ASUSP	Suspend all the tasks with a specified priority-number
BACK	Restore a swapped program
BCLR	Clear a bit of a word to zero
BOOT	Disk bootstrap
BSET	Set a bit of a word to one
CDIR	Create a sub-directory with a specified name
CFILW	Create RDOS disk file
CHAIN	Overwrite current program with a program from disk
CHLAT	Change, add or delete a link access attribute
CHNGE	Change priority number of a specified task
CHRST	Restore channel status to re-read/re-write the same record
CHSAV	Save channel status to re-read/re-write the same record
CHSTS	Get a copy of UFD of a file specified by a channel
CLOSE	Close a file
CPART	Create a secondary partition
DATE	Get date
DELETE	Delete a file
DFILW	Delete RDOS disk file

Table 1. Real Time Facilities (Cont'd)

Name	Function
DIR	Change current default directory device
DLINK	Define a link entry in current directory for a file in other directory
DULNK	Define a user clock Delete a link entry of current directory
EBCK	Restore a swapped program or return to CLI if no swapped program exists
EQUIV	Add a new name to multiple file device
EXBG	Execute a program in background
EXFG	Execute a program in foreground
EXIT	Terminate a task
FBACK	Restore a swapped program
FCHAN	Overwrite current program with a program from disk
FCLOS	Close a file of a specified channel and free the channel
FDELY	Suspend a task during specified time
FGND	Know whether a foreground program is executable or not
FGTIM	Get time of day
FINRV	Disconnect a user interrupt device from system
FINTD	Find a device which is requesting interruption
FOPEN	Assign a channel to a file (device)
FOVLD	Load an overlay in multi-task environment
FOVRL	Release an overlay
FQTASK	Execute a task or an overlay periodically
FSEEK	Position to a specified record of a random file
FSTAT	Set or change a file attribute
FSTIM	Set a real time clock
FSWAP	Swap in/swap out
FTASK	Generate a task

Table 1. Real Time Facilities (Cont'd.)

Name	Function
GCIN	Get current input device name
GCOUT	Get current output device name
GDIR	Get current default directory/device name
GFREQ	Get frequency of real-time clock
GSYS	Get current system name
GTATR	Get file attributes
HOLD	Hold a specified task
ICLR	Clear a bit of a word to zero
ICMN	Define an area to send and receive message
INIT	Initialize a directory
ISET	Set a bit of a word to one
ITASK	Generate a task and give ID to it
KILL	Terminate an executing task
MDIR	Get current master device name
MTDIO	Operate MT or cassette on machine level
MTOPD	Open MT or cassette by free format I/O
ODIS	Disable interruption from console
OEBL	Enable interruption from console
OPEN	Open a file
OVERFLOW	Check overflow of floating point
OVEXT	Release an overlay and return control to return-location
OVEXX	" " "
OVKIL	Release an overlay and terminate a task containing it
OVKIX	" " "
OVLOD	Load an overlay in single-task environment
OVOPN	Open an overlay file

Table 1. Real Time Facilities (Cont'd)

Name	Function
PRI	Change priority number of executing task
RDBLK	Read specified numbers of blocks from a consecutive file or a random file
RDCMN	Read a message from other program communication area
RDOPR	Read an operator message
RDRW	Read specified numbers of records from a file
READR	" "
REC	Receive a message
RELSE	Make a specified task ready
RENAM	Rename a disk file
RESET	Close all open files
RLSE	Close and release all files of a specified directory
RUCLK	Release a user clock
SDATE	Set date
SPDITE	Disable spooling of a specified device
SPEBL	Enable spooling of a specified device
SPKIL	Stop current spooling operation
START	Execute a task after specified time delay
STAT	Get current status of a specified file
STIME	Set time
STTSK	Get current status of a task
SUSP	Suspend a task
SWAP	Swap in/swap out
TIME	Get current time of day
TRNON	Execute a task at a specified time of day
UPDATE	Update the size information of current file

Table 1. Real Time Facilities (Cont'd)

Name	Function
WAIT	Suspend a task during specified time
WRBLK	Write the content of an array by block into a disk file specified by channel
WRCMN	Write a message into other program communication area
WRITR	Write some records into a file specified by channel
WROPR	Write an operator message
WRTR	Write some records into a file specified by channel
XMT	Send a message to other task
XMTW	Send a message to other task and suspend its own execution

Table 2. Process I/O Facilities

Following subroutines are referenced by CALL statement:

Name	Function
DOPEN	Initialize a device or a library
DCLOS	Close a device or a library
PSHAD	Create and set hard image parameters to a device
IPSHAD	Set hard image data sampling conditions to a device
GPHAD	Get data through PIO in non-interruptable state
GMHAD	Get data in seqnential mode in non-interruptable state
GDHAD	Get data in sequential mode in interruptable state
DSHAD	Stop cyclic DMA transfer at optional timing
PCONV	Create hard image parameters in batch mode

11. Takeda Riken Mfg. Co.

1. Language Name

   Lab, BASIC

2. Hardware

   TACC-1200 (Equivalent to NOVA-1200 of Japan Mini-Comp Co.)

3. Base Language, Level and Design Objective

   Base language : BASIC

   Level : Dartmouth BASIC

   Design objective : 1) Real time data collection
   2) Conversational programming
   3) BASIC for industry application

4. Character Set, Label, Variable Name

   Character set : 36 alphanumerals

   Label : Unsigned integer number (1 ~ 9999)

   Variable name : A single letter of the alphabet or a single letter of the alphabet followed by one of the digits 0 through 9

5. Data Types and Precision

   Real (including integer) : 8 significant digits (precision : max 8 decimal digits)
   $\pm 2.0 * 10^{-39} \sim \pm 1.7 * 10^{38}$

   Character : ASCII

   Octal number

6. Default Rules

   None

7. Storage Allocation

   Static

8. Common Variables

   All are common variables

9. Variables on External Memory

   None

10. Array Dimension

2 dimensions

11. Structure

None

12. Built-in Functions

See attached Table 1.

13. Language Expansion Capability or Conditional Compile/Macro Facility

None

14. Character/Bit String Manipulation Capability

None (Logical operation is available. See item 22)

15. Interruption Handling

None

16. Input/Output

None

17. File Organization

None

18. Method of Input/Output from/to I/O Unit (TW, CRT, etc.)

1) Availability of READ/WRITE Statements

None

2) Others

Provides I/O statements suitable for XY plotter and operator console. Data specification and format specification are defined in arguments of these statements.

19. Debug Facility

There is no statement for debug but is facility to set as keyboard input command in prior to program execution.

1) Debugging support facility

RENUMB : Renumber statement number of a program

TRACE

BREAK : Set break points

2) Flowchart generator

3) Dynamic debugging facility

Modify statement of a specified line or confirm the content of a specified line at execution-time.

20. Real Time Facilities

See attached Table 2.

21. Process I/O Facilities

See attached Table 3.

22. Other Facilities

1) Assembler program linkage facility

PRO	Describe assembler as a statement
RPRO	Read assembler
NAME	Regist a keyword for a specified assembler program (subroutine)

2) Logical operation facility for octal data

23. Applicable Area

Scientific application, data collection and analysis system, chemical data processing system, automatic measurement/check system, IC tester control, data processing system, process monitor, data logging system, etc.

Table 1. Built-in-Functions

Following functions are available through LET statement:

Name	Function
ABS (X)	Absolute value of X
EXP (X)	e^X
INT (X)	Integral part of X
LOG (X)	$\log e^X$
RND (X)	Random number between 0 and 1
SGN (X)	Sign of X (-1, 0, or +1)
SQR (X)	$\sqrt{X}$
SIN (X)	Sine of X radians
COS (X)	Cosine of X radians
TAN (X)	Tangent of X radians
ATN (X)	Arctangent (in radians) of X

Table 2. Real Time Facilities

Following service routines are embedded in the interpreter and they are described as one of program statements:

Name	Function
TASK	Specify priority of a task and starting statement number
TEND	Terminate a task
TRNON	Initiate a specified task at a specified time of day
START	Initiate a specified task after specified time delay
WAIT	Suspend its own execution during specified time
DSABLE	Suspend a specified task
ENABLE	Make a suspended task ready state
STRIKE	Activate a specified task by a program

Table 3. Process I/O Facilities

Following service routines are embedded in the interpreter and they are described as one of program statements:

Name	Function
AISQV	Read analog voltage in sequential order
AIRDV	Read analog voltage in random order
DVM	Read a measured data from a digital voltmeter (-TR-7304)
DMM	" " (-TR-6656)
IDV	" " (-TR-6567)
FDVM	Set a measurement condition to a digital voltmeter (-TR-7304)
FDMM	" " (-TR-6656)
FIDV	" " (-TR-6567)
DIW	Read digital input data
DOW	Write digital output data
AOV	Analog output of digital value to a specified channel
EVHNT	Execute a corresponding task at the time of event occurrence
TSET	Set system clock
TSTOP	Stop system clock
TIME	Read system clock

12. Takeda Riken Mfg. Co.

1. Language Name

MOST

2. Hardware

NOVA (Model 01)

3. Base Language, Level and Design Objective

Base language : FORTRAN, BASIC, PL/I

Level : ——

Design objective : Language for test and evaluation of digital semi-conductor elements

4. Character Set, Label, Variable Name

Character set : 36 alphanumerals

Label, Variable Name : 1 ~ 6 alphanumerals, where the first must be alphabet

5. Data Types and Precision

Integer/Real : Magnitude $10^{-16} \sim 10^{15}$

Precision $2^{-21} \sim 2^{21}-1$

Constants :

Logical	16 bits string	
Arithmetic	Constant without unit	
Current	Having units of	A(ampere), M(milli), U(micro), N(nano)
Resistance	"	R(ohm), K(kilo), M(mega), G(giga)
Voltage	"	V(volt), M(milli), U, N
Time	"	S(second), M(milli), U, N
Frequency	"	HZ(hertz), K, M(mega), G
Character	Internal code (ASCII)	

6. Default Rules

None

7. Storage Allocation

Static

8. Common Variables

None

9. Variables on External Memory

None

10. Array Dimension

3 dimensions

11. Structure

None

12. Built-in Functions

See attached Table 1.

13. Language Expansion Capability or Conditional Compile/Macro Facility

None

14. Character/Bit String Manipulation Capability

Available.	Character string	:	up to 2 characters
	Bit string	:	up to 16 bits

15. Interruption Handling

None

16. Input/Output

READ/WRITE statements

17. File Organization

Sequential file

18. Method of Input/Output from/to I/O Unit (TW, CRT, etc.)

1) Availability of READ/WRITE Statements

WRITE statement only is available. For input, INPUT statement is used. Linkage is made to the handler of the monitor at execution time.

2) Others

None

19. Debug Facility

None. However, a system program for debug purpose is provided.

20. Real Time Facilities

Unknown

21. Process I/O Facilities

Unknown

22. Other Facilities

Unknown

23. Applicable Area

Scientific application, LSI test system, LSI production line control, etc.

Table 1. Built-in Functions

Name	Function
BITX	Reverse bit string
SR, SL	Shift bit string
SQRT	Square root
RANDOM	Random number
ABS	Absolute value
MAX	Largest value
MIN	Smallest value
CODE	Find a constant class
SIGN	Polarity
UNIT	Change a class of a constant
RANK	Rand a data by the value
TRUNK	Fix
FLOOR	
COVE	
CEIL	

13. Yokogawa Electric Works Co.

1. Language Name

YOS-MD FORTRAN, YOS-LD FORTRAN

2. Hardware

YODIC 100, YODIC 1000

3. Base Language, Level and Design Objective

Base language	:	FORTRAN
Level	:	JIS FORTRAN (Level 5000) + α
Design objective	:	1) FORTRAN extensions of data types and operations used for process control
		2) Facility of in line assembling and linkage to assembler programs
		3) Optimization of object program

4. Character Set, Label, Variable Name

Character set	:	36 alphanumerals, 18 special characters
Label	:	Unsigned integer number (1 ~ 99999)
Variable name	:	1) 1 ~ 6 alphanumerals, where the first must be alphabet
		2) 2 ~ 6 alphanumerals, where the first must be = or &, and the second must be alphabet
		3) 2 ~ 7 alphanumerals, where the first must be @, <or> , and the second must be alphabet

5. Data Types and Precision

Integer	:	Single (16 bits)	$-2^{15} \sim 2^{15}-1$	
		Double (32 bits)	$-2^{30} \sim 2^{30}-1$	
Real	:		Precision	Magnitude
		Single (32 bits)	6.6 decimal digits	$2^{-128} \sim 2^{127}$
		Double (64 bits)	15.7 decimal digits	$2^{-128} \sim 2^{127}$
Logical				
Character	:	Internal code (ISO code)		
Bit	:	1 ~ 16 bits, available in arithmetic expression		
Hexadecimal				

Binary : max 16 beginning with B

Mask data : Available by built-in function IXOR

Address

6. Default Rules

- Type declaration: A variable whose first character is I~N, =, &, or @ is interpreted as integer and A~H, O~Z, > , or < is as real

- An array name appeared in DATA statement and I/O list specifies all of the array elements in the array.

- Increment parameter of DO statement : Default value is 1

- Format of READ/WRITE statement : Data I/O without data conversion

- Array subscript of an array whose name begins with @, >or <: The first element of the array

7. Storage Allocation

Static

8. Common Variables

- Core resident variable : Labeled common block, or variable name beginning with =

- Process variable : Variable name beginning with @, >or<

9. Variables on External Memory

Variable name beginning with @, >or<

10. Array Dimension

3 dimensions

11. Structure

Available by PACK DATA statement.

12. Built-in Functions

All of intrinsic functions and basic external functions defined in JIS FORTRAN (Level 7000) are available except REAL, CMPX, CONJG, CABS, CEXP, CLOG, CSIN, CCOS and CSQRT. In addition to them, provided are LABS, LMAX, LMIN, FLOATL, DFLOAT, LFIX, LFIXD, ISNGL, LDBLE, IAND, IOR, NOT, IEOR, ISHIFT and IXOR.

13. Language Expansion Capability

1) Conditional compile : Capability to specify at compile time *foreground* task or background task, optimization, and inclusion of debug statements

2) Macro capability : Available by means of inline assembling facility

3) A variable whose symbolic name begins with @, > or < may be expanded or changed to be treated as common variable among task, not only as process variable.

14. Character/Bit String Manipulation Capability

Bit string manipulation is available by means of the built-in functions of IAND, IOR, IEOR, ISHFT and IXOR.

15. Interruption Handling

None

16. Input/Output

1) READ/WRITE statement : I/O for drum, disk and cassette tape, using buffering

2) FGET/FPUT, DGET/DPUT subroutine : Output to drum and disk. Buffering is specified by parameters

17. File Organization

- Drum and disk : Undefined file, indexed sequential file

- Cassette tape : Files peculiar to cassette tape

18. Method of Input/Output from/to I/O Unit (TW, CRT, etc.)

1) Availability of READ/WRITE Statements

Input/Output to/from TW and CRT is described by READ/WRITE statements based on FORTRAN. Special format specification for TW and CRT is specified as extended field description of FORMAT statements.

2) Others

CALL subroutines are provided for I/O units.

19. Debug Facility

TRACE VALUE

TRACE CONTROL

TRACE CALL

CHECK SUBSCRIPT

RENAME

20. Real Time Facilities

See attached Table 1.

21. Process I/O Facilities

See attached Table 2.

22. Other Facilities

1) Segmentation : Described in a program by DEFINE MODULE statement

2) Overlay structure :
- Specified in a program
- Specified by control statement at the time of task registration
- Automatically specified by compiler

3) Multi program entries for a main program and for a subprogram

4) Task entries : Max. 16 for a task

5) Multi-file capability for cassette tape

6) Reference of variables and arrays among programs :
By EXTERNAL SYMBOL statement, ENTRY SYMBOL statement

7) Multiple return point from a subroutine/subprogram

8) System subroutine : May compile as system subroutine (relocatable object) by SUBMAIN statement

9) Mixed mode expression

10) Option return in READ/WRITE statement:

READ (u, f, END=n1, ERR=n2) K
WRITE (u, f, END=n1, ERR=n2) K
END=n1, and/or ERR=n2 may be omitted.
n1 is next executable statement when I/O is completed, and n2 when an I/O error occurs.

Table 1. Real Time Facilities

Following service routines are referenced by CALL statement:

Name	Function
IREQ	Initiate a specified task
CTLPT	Work or stop a program clock By specifying Interval Timer: Execute a specified task after specified time delay By specifying Periodic Timer: Execute a specified task at the intervals of specified time
CTLHT	Work or stop a hardware clock. The functions of the facility is the same as CTLPT
BRSTK	Control the execution of multi-entry-task and pass the control to the specified entry point
DEF	Enable an event occurrence
WAIT	Make a task wait state until an event specified by DEF occurs and is posted
POST	Post occurrence of an event specified by DEF
OPEN	Declare the use of a resource. When the resource was already OPENed by other task, the task is "WAIT" until the other task CLOSE the resource
CLOSE	Declare the termination of use of a resource
LOAD	Transfer a segment (task segment or system subroutine) from auxiliary storage to main memory

Table 2. Process I/O Facilities

Following service routines are referenced by CALL statement:

Name	Function
ASCAN	Read analog input data in sequential order. At the time of completion, able to execute a specified task or post an event occurrence by POST subroutine.
AOUT	Handle direct mode analog output in random mode
CIN	Handle direct mode digital input in random mode
CINS	Status input with image
COUT	Handle direct mode digital output in random mode. Output status is retained until changed by request. There is an option to output a reset signal automatically after specified time of delay.
PIN	Handle pulse input in random mode
PSCAN	Handle pulse input specified by a series of parameters. Same as reiterated PIN.
POUT	Handle pulse output in random mode
OSOUT	Write differential output value to a specified output station through analog control output module
OMOUT	Write positional or differential output value to analog control output module. Able to read the value before output at the same time.

14. Mitsubishi Electric Mfg. Co.

1. Language Name

Realtime FORTRAN

2. Hardware

MELCOM 350-7

3. Base Language, Level and Design Objective

Base language	:	FORTRAN
Level	:	JIS FORTRAN (Level 7000)
Design objective	:	- Language suitable for industrial application
		- Executable under real time monitor

4. Character Set, Label, Variable Name

Character set	:	36 alphanumerals, 14 special characters
Label	:	Unsigned integer number (1 ~ 3276)
Variable name	:	1 ~ 6 alphanumerals, where the first must be alphabet.

5. Data Types and Precision

Integer	:	Single (16 bits)	$-2^{15} \sim 2^{15}-1$	
Real	:		Precision	Magnitude
		Single (32 bits)	6 decimal digits	$10^{-78} \sim 10^{75}$
		Double (64 bits)	16 "	"
Logical	:	Available, word type and bit type		
Character	:	Internal code (ASCII code)		
Bit	:	Available as a value		
Complex	:	Single real number		
Hexadecimal number				

6. Default Rules

Type declaration : I - N rule

Array subscript : The first element of the array

7. Storage Allocation

Static

8. Common Variables

Labeled common block

9. Variables on External Memory

None

10. Array Dimension

3 dimensions

11. Structure

None

12. Built-in Functions

All of intrinsic functions and basic external functions defined in JIS FORTRAN (Level 7000) are available.

13. Language Expansion Capability or Conditional Compile/Macro Facility

None

14. Character/Bit String Manipulation Capability

None

15. Interruption Handling

Available

16. Input/Output

READ/WRITE subroutine (with buffer)

17. File Organization

Sequential file, Random file

18. Method of Input/Output from/to I/O Unit (TW, CRT, etc.)

1) Availability of READ/WRITE Statements

Available. Link to an execution time I/O system, FONATTER

2) Others

X-Y plotter : CALL type

19. Debug Facility

Trace : Statement label, variable name, value of a variable, subscripted value, condition of IF statement.

20. Real Time Facilities

See attached Table 1.

21. Process I/O Facilities

See attached Table 2.

22. Other Facilities

1) Inline assembly facility:

- Assembler language can be intermingled with FORTRAN statements.
- Variable names of a FORTRAN program can be referred by assembler instruction.

2) Segmentation facility :

Specify segment overlay in a program (by CALL statement).

23. Applicable Area

Process control in steel, power, petroleum, chemistry, cement, wood pulp and food industries.
Data collection and system control in water processing, building control and power transmission/distribution areas.

Table 1. Real Time Facilities

Following subroutines are referenced by CALL statement:

Name	Function
Standard subroutines	
M : AB	Make a specified task executable
M : IN	Initiate an executable task
M : DA	Suspend itself
M : EX	Terminate itself
Option subroutines	
M : HT	Terminate a specified task .
M : ER	Print out error message
M : TD	Suspend itself during specified time
SF: AB	Make a specified sub-task executable
SF: IN	Initiate an evecutable sub-task
SF: DA	Suspend itself (sub-task)
SF: EX	Terminate itself
SF: CH	Overlay loading of a specified program
M : SGLD	Overlay loading of a specified segment (a part of a program)

Table 2. Process I/O Facilities

Following subroutines are referenced by CALL statement:

Name	Function
DI	Digital input
DIW	Digital input. Not return to a calling point until the completion of input operation
DOM	Digital output momentary
DOMW	Same as DOM. Not return to a calling point until the completion of output operation
DOL	Digital output latching
DOLW	Same as DOL. Not return to a calling point until the completion of output operation
AI	Random analog input
AIW	Same as AI. Not return to a calling point until the completion of input operation
AO	Random analog output
AOW	Same as AO. Not return to a calling point until the completion of output operation

15. Mitsubishi Electric Mfg. Co.

1. Language Name

   CONFORM - TV

2. Hardware

   MELCOM 350-30F/50

3. Base Language, Level and Design Objective

   Base language : FORTRAN

   Level : JIS FORTRAN (Level 7000)

   Design objective : - Easy to use

   - High performance process control software

4. Character Set, Label, Variable Name

   Character set : 36 alphanumerals, 13 special characters

   Label : Unsigned integer number (1 ~ 32767)

   Variable name : 1 ~ 6 alphanumerals or $, where the first must be alphabet or $

5. Data Types and Precision

Integer :	Single (16 bits)	$-2^{15} \sim 2^{15}-1$	
	Double (32 bits)	$-2^{31} \sim 2^{31}-1$	
Real :		Precision	Magnitude
	Single (32 bits)	6 decimal digits	$10^{-76} \sim 10^{75}$
	Double (64 bits)	16 decimal digits	"

   Logical : Available (word type and bit type

   Character : Internal code (EBCDIC code)

   Bit : Can be used as variable

   Complex : Single real number

   Pointer variable

   Hexadecimal number

6. Default Rules

Type declaration : I - N rule

Array subscript : The first element of the array

7. Storage Allocation

Static, Based

8. Common Variables

GLOBAL type variable

9. Variables on External Memory

None

10. Array Dimension

3 dimensions

11. Structure

None

12. Built-in Functions

All of intrinsic functions and basic external functions defined in JIS FORTRAN (Level 7000) are available and additionally ADDR (address) is provided.

13. Language Expansion Capability or Conditional Compile/Macro Facility

Conditional compile facility

14. Character/Bit String Manipulation Capability

Logical operation for single integer number is available.

15. Interruption Handling

None

16. Input/Output

FETCH/STORE statement (with buffer)

AD-A068 267 PURDUE UNIV LAFAYETTE IND PURDUE LAB FOR APPLIED IND--ETC F/G 9/2
MINUTES 1978 SPRING REGIONAL MEETING INTERNATIONAL PURDUE WORKS--ETC(U)
JUN 78 N00014-78-C-0127

UNCLASSIFIED NL

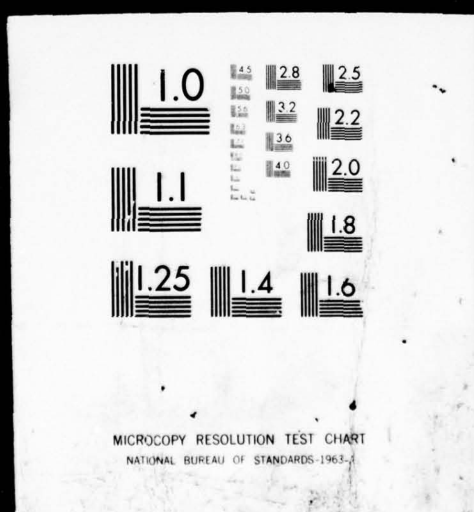
2 OF 4

AD
A068267

IFIED

2 OF 4

AD A068267



17. File Organization

Random access

Sequential

(There is no concept of file.)

18. Method of Input/Output from/to I/O Unit (TW, CRT, etc.)

1) Availability of READ/WRITE Statements

Available. The compiler generates linkage sequences to the execution-time I/O writer.

2) Others

None

19. Debug Facility

Trace facility

20. Real Time Facilities

See attached Table 1.

21. Process I/O Facilities

PROCES-I/O statements are provided and P I/O functions of OS are processed through these statements.

22. Other Facilities

1) Inline assembly facility :

- Assembler language can be intermingled with CONFORM statements
- Assembler language parts and CONFORM statements can refer to data mutually.

2) Segmentation facility:

- Entire or a part of a program are overlayable
- A variable common only among segments can be declared

3) Internal data conversion facility

4) Reentrant subroutines

5) Multi-entry facility

6) Multi-return facility

7) Internal subroutine facility

23. <u>Applicable Area</u>

Production control, inventory control, economical load balancing and experimental data collection in power, gas, steel, chemistry, petrolic chemistry, petroleum refining, cement, paper, pulp and food industries. Non-manufacturing process of water processing, etc.

Table 1. Real Time Facilities

They are described as statement.

Statement Name	Function
ASSIGN - TASK	Define a group name of tasks
CALL - TASK	Initiate a specified task
WAIT	Terminate an executing task
DELAY	Suspend a specified task during specified time
PURGE	Force to halt a specified task
READ - TIMER	Read current time of day

16. Tokyo Shibaura Electric Co.

1. Language Name

PL/40

2. Hardware

TOSBAC - 40 (40D, 40C, 40L)

3. Base Language, Level and Design Objective

Base language	:	PL360, ALGOL
Level	:	System description language
Design objective	:	Language for system description and application description for mini-computers

4. Character Set, Label, Variable Name

Character set	:	36 alphanumerals, 18 special characters
Identifier	:	1 ~ 12 alphanumerals (including $), where the first must be alphabet

5. Data Types and Precision

Integer	:	Single (16 bits)	$-2^{15} \sim 2^{15}-1$
		Double (32 bits)	$-2^{31} \sim 2^{31}-1$
Real	:	Single (32 bits)	
		Double (64 bits)	

Decimal number : Available by FUNCTION

Character : ASCII code

Bit : None. Set, reset or test of bits is available

Hexadecimal number

Address data

6. Default Rules

Identifiers including labels and internal subroutines must be declared without omission.
Identifiers defined as array can be referred without subscripts.

7. Storage Allocation

Static

8. Common Variables

External identifier enclosed by LINK ~ KNIL.

9. Variables on External Memory

None

10. Array Dimension

1 dimension

11. Structure

None

12. Built-in Functions

The compiler does not have its own subroutines, but may refer to subroutines of TOSBAC F40 FORTRAN and others.

13. Language Expansion Capability or Conditional Compile/Macro Facility

- Selective (conditional) compile facility
- Capability to use assembler instructions by FUNCTION

14. Character/Bit String Manipulation Capability

Character string is available through FUNCTION.

15. Interruption Handling

Interruption is detected by hardware and control is passed to internal routines of OS.

16 ~ 18. Input/Output, File Organization

Because of system description language, there is not such high level I/O facility that requires run-time subroutines.
Able to use FORTRAN subroutines of TOSBAC-40 and OS macros by FUNCTION.

19. Debug Facility

Facility to set selective compiling portions.

20. Real Time Facilities
21. Process I/O Facilities

} Able to use T-40 FORTRAN subroutines.

22. Other Facilities

1) Specification of a program entry by ENTRY declaration (ENTRY PROC)

2) Identifiers are externally referrable.

3) Block structures of BEGIN ~ END and PROC ~ END.

4) Compound statements e.g. IF ~ THEN ~ ELSE ~ FI;
CASE ~ OF ~ ESAC,
REPEAT ~ UNTIL ~ TAEPER; etc.

5) Linkage with assembler language.

6) Comments may be written anywhere in a program

23. Applicable Area

System description for systems of T-40 minicomputer (such basic software as scientific application, process control, language processor, etc.)

17. Tokyo Shibaura Electric Co.

1. Language Name

Extended Industrial FORTRAN

2. Hardware

TOSBAC-40D (TOSBAC-40C, 40L)

3. Base Language, Level and Design Objective

Base Language	:	FORTRAN
Level	:	JIS FORTRAN (Level 7000)
Design Objective	:	Compact compiler for minicomputers

4. Character Set, Label, Variable Name

Character set	:	36 alphanumerals, 10 special characters
Level	:	Unsigned integer number (1 ~ 99999)
Variable name	:	1 ~ 6 alphanumerals, where the first must be alphabet.

5. Data Types and Precision

Integer	:	Single (16 bits)	$-2^{15} \sim 2^{15}-1$	
		Double (32 bits)	$-2^{31} \sim 2^{31}-1$	
Real	:		Precision	Magnitude
		Single (32 bits)	6.3 decimal digits	$10^{-76} \sim 10^{75}$
		Double (64 bits)	15.9 decimal digits	$10^{-76} \sim 10^{75}$

Complex : Single precision

Logical

Hexadecimal number : Available only in DATA statement and FORMAT statement.

Character : Available only in DATA statement and FORMAT statement. Can use ASCII code and KATAKANA

6. Default Rules

Type declaration : I - N rule

7. Storage Allocation

Static

8. Common Variables

Labeled common block and blank common block
GLOBAL type variable

9. Variables on External Memory

None

10. Array Dimension

3 dimensions

11. Structure

None

12. Built-in Functions

Intrinsic functions and basic external functions defined in JIS FORTRAN (Lavel 7000) are available except DMAX1 and CABS. In addition to then, provided are IDABS, IDDINNT, IDMOD, DMAXO, IDMINO, DFLOAT, IDSIGN, IDSNGL and IDBLE.

13. Language Expansion Capability or Conditional Compile/Macro Facility

Capability of JCL specifying valid/invalid of trace function.

14. Character/Bit String Manipulation Capability

None

15. Interruption Handling

None (Interruption is detected by hardware and control is passed to OS routines)

16. Input/Output

1) READ/WRITE statements (MT/CMT)

2) REWIND statement, BACKSPACE statement, BACKFILE statement (MT/CMT), SKIPRECORD statement, SKIPFILE statement, ENDFILE statement

3) BULKIN/BULKOT subroutine (DISK/DRUM)

4) ADDIN subroutine (color-CRT), RAM/RAMH subroutine

17. File Organization

Sequential file, Random file

18. Method of Input/Output from/to I/O Unit (TW, CRT, etc.)

1) Availability of READ/WRITE Statements

Available. READ/WRITE statements are also available for color CRT.

2) Others

- A FORTRAN subroutine which functions as a utility for GRAPHIC.

- Data transmission by WRITE statement (with FORMAT)

19. Debug Facility

1) Trace facility : TRACE ON
TRACE ON*
TRACE OFF

2) A simulator-type-debugger is provided outside language processor

20. Real Time Facilities

See attached Table 1.

21. Process I/O Facilities

See attached Table 1.

22. Other Facilities

1) Segmentation : Subroutine-type FTLINK and RTLINK

2) Intermix with assember : ADMIX

3) Numerical analysis library and statistical library are provided as FORTRAN subroutines.

23. Applicable Area

Scientific application, process control, etc.

Table 1. Real Time Facilities and Process I/O Facilities

Name	Function
TRNON	Request to initiate a specified program at a specified time
START	Request to initiate a specified program after specified time delay
WAIT	Make a program wait during specified time
DIW	Digital input in any order according to a specified group address
DOLW	Digital output in any order according to a specified point address
DOMW	Retain digital signals during specified time
RO	Relay output to a specified controller/group address
AISQW	Input analog points data in sequential order from specified point address
AIRQW	Input analog points data according to a specified point address
AOW	Output analog points data in sequential order from a specified point address
PO	Output a specified pulse count or time duration to a specified controller/point address

18. Tokyo Shibaura Electric Co.

1. Language Name

   TPL

2. Hardware

   TOSBAC - 40 (40C, 40L)

3. Base Language, Level and Design Objective

   Base language : PL/I (BASIS/1-12)

   Level : Subset + α

   Design objective : Compact PL/I subset for mini-computer

4. Character Set, Label, Variable Name

   Character set : 36 alphanumerals, 19 special characters

   Label / Variable name (includes constant name) : 1 ~ 12 alphanumerals, where the first must be alphabet

5. Data Types and Precision

Integer	:	Declared precision 1 ~ 15	Single (16 bits)	$-2^{15} \sim 2^{15}-1$
		16 ~ 31	Double (32 bits)	$-2^{31} \sim 2^{31}-1$
Real	:	Declared precision 1 ~ 21	Single (32 bits)	
		22 ~ 53	Double (64 bits)	

   Logical : Available as a result of evaluation

   Character : ASCII (1 ~ 255 characters)

   Bit : 1 ~ 16 bits

   Pointer Variable

   Label constant/entry constant

   Format variable/constant

   Others : File constant, task constant, semaphore constant, etc.

6. Default Rules

- A name which is not declared explicitly nor contextually is interpreted as variable of single precision integer

- An array name without subscript is interpreted as the entire array

7. Storage Allocation

Static, Based, Automatic

8. Common Variables

EXTERNAL variable

GLOBAL variable

9. Variables on External Memory

None

10. Array Dimension

3 dimensions

11. Structure

Available. 3 levels and 1 dimension array per level

12. Built-in Functions

Intrinsic functions and basic external functions defined in JIS FORTRAN (Level 7000) are available except IFIX, SNGL, REAL, AIMAG, DBLE, CMPLX, CONJG, CEXP, CLOG, CSIN, CCOS, CSQRT and CABS. Additionally, following functions are available : FIXED, BIT, PREC (which are data type conversion), ADDR, IDABS, IDDINT, IDMOD, IDMAXO, IDMINO, IDSIGN and AGCMP (aggregate compare).

13. Language Expansion Capability or Conditional Compile/Macro Facility

Conditional compiling of debug statements

14. Character/Bit String Manipulation Capability

Available

15. Interruption Handling

Interruption is detected by hardware and control is passed to programs of operating system

16. Input/Output

    1) Named file by READ/WRITE and OPEN/CLOSE statements

    2) BULKIN/BULKOT CALL subroutine

17. File Organization

    Sequential file, Random file

18. Method of Input/Output from/to I/O Unit (TW, CRT, etc.)

    1) Availability of READ/WRITE Statements

       Available. No format conversion

    2) Others

       GET/PUT statements : with format conversion

19. Debug Facility

    1) DEBUG statement : Trace

    2) PROTECT statement : Protect the value of a parameter or a variable (compile time facility)

    3) Others : Compile time process of the introduction of CONSTANT attribute and type check of CALL argument.

20. Real Time Facilities

    See attached Table 1.

21. Process I/O Facilities

    Same as those of Toshiba Extended FORTRAN

    (See Table 1 of Section 17)

22. Other Facilities

    1) A constant may have a name

       e.g. DCL PAI FLOAT (21) CONSTANT (3.1416);

    2) Program blocking by BEGIN ~ END, PROC ~ END and DO ~ END

    3) Compound statements : IF ~ THEN ~ ELSE
       CASE ~ OF ~ OTHERWISE

4) LIST type GET/PUT statements

5) Free format description and comments are permitted anywhere

6) Option of RECURSIVE/REENTRANT program

23. <u>Applicable Area</u>

Process control, scientific application, etc.

Table 1. Real Time Facilities

Following statements are provided for real time purpose:

Name	Function
TRNON	Initiate a task at a specified time of day
START	Initiate a task after specified time delay
LINK	Linkage among tasks
DELAY	Delay a task
STOP	Terminate a task
WAIT	Wait the completion of an event
REQUEST	Request semaphore
RELEASE	Release semaphore

19. Sinko Electric Co.

1. Language Name

    FORTRAN

2. Hardware

    SCCS77

3. Base Language, Level and Design Objective

Base language	:	FORTRAN
Level	:	Approximately JIS FORTRAN Level 5000/7000
Design objective	:	Compiler for both scientific and industrial applications

4. Character Set, Label, Variable Name

Character set	:	36 alphanumerals, 12 special characters
Label	:	Unsigned integer number (1 ~ 32767)
Variable name	:	1 ~ 6 alphanumerals, where the first must be alphabet

5. Data Types and Precision

Integer	:	Single (16 bits)	$-2^{15} \sim 2^{15}-1$
Real	:	Single (32 bits)	Magnitude $10^{-76} \sim 10^{75}$ Precision 6.3 decimal digits
Logical			
Character	:	Internal code (ASCII code) /	
Bit	:	By means of bit operation functions	

6. Default Rules

    Type declaration : I-N rule

7. Storage Allocation

    Static

8. Common Variables

GLOBAL

9. Variables on External Memory

Unknown

10. Array Dimension

3 dimensions

11. Structure

None

12. Built-in Functions

Intrinsic functions and basic external functions defined in JIS FORTRAN (Level 7000) are available except REAL, AIMAG, CMPLX, CONJG, CEXP, CLOG, CSIN, CCOS, CSQRT and CABS.
Additional builtin functions for bit manipulation are provided: IOR, IAND, INOT, IXOR and ISHIFT.

13. Language Expansion Capability or Conditional Compile/Macro Facility

None

14. Character/Bit String Manipulation Capability

None

15. Interruption Handling

Available : Link with assembler

16. Input/Output

READ/WRITE statements

17. File Organization

Sequential file

18. Method of Input/Output from/to I/O Unit (TW, CRT, etc.)

1) Availability of READ/WRITE Statements

Available

2) Others

None

19. Debug Facility

TRACE IN : Virtual input instruction

TRACE OUT : Virtual output instruction

20. Real Time Facilities

See attached Table 1.

21. Process I/O Facilities

See attached Table 2.

22. Other Facilities

None

23. Applicable Area

Scientific application, process control, instrumentational application, production line management, equipment control, system control, etc.

Table 1. Real Time Facilities

Following subroutines are referenced by CALL statement:

Name	Function
REQ	Initiate a specified task
TIM	Initiate a specified task after specified time delay
TIM	Initiate a specified task after specified time delay and repeat it with the same time-interval.
IOWAIT	Declare to become wait state until completion of I/O operation
ERR	Pass information to error processing procedure when error occurs
WAIT	Become wait state and pass control to other executable task
SED	Enquene for resources (program or data)
RSED	Dequene for resources
ETO	Terminate abnormally the task itself or other tasks

Table 2. Process I/O Facilities

Following service routines are referenced by control statement:

Name	Function
DI	Read 1-bit ON/OFF input Read n-bits ON/OFF input Read n x m bits and recognize the pattern Read n digits of BCD data and convert to a specified format
DO	Write 1-bit ON output Write 1-bit OFF output Write n-bits output Write 1-bit ON/OFF output m times Write data with BCD format by converting from code type, integer type and floating point type
PI	Read pulse values accumulated in 1-bit or 12-bit counter
AI	Convert analog input with a specified scale
AO	Convert analog output with a specified scale
WDT	Reset watch-dog timer

APPENDIX J-III

TC-5

INTERFACES AND DATA TRANSMISSION COMMITTEE

PURDUE JAPAN

1. Tohyama, Takashi, Industrial Dataway in Distributed Control System, Paper presented at IFAC Congress, Helsinki, Finland, June 1978.

INDUSTRIAL DATAWAY IN DISTRIBUTED CONTROL SYSTEM

Takashi Tohyama

Chiyoda Chemical Engineering & Construction Co.
Yokohama, JAPAN

Abstruct. The paper reviews data communication technogy to be used to interconnect both process control and process computer apparatus to the assembled instrumentation and control system. It describes a design guideline for the implementation of industrial process computer inter-subsystem communication. The advent of the micro computer facilitates the use of distributed control system consisting of intellegent subsystems such as multiloop controllers, process operator's consoles, and dedicated computer system. Herein, the industrial process computer inter subsystem communication means (hereafter called the Industrial Dataway) is a key subject to be standerdized in order to reduce complexity and costs, and to improve interchangability and flexibility. The activities of the Japanese working group for standardization related IEC/TC65A/WG6, and a recommended design guideline proposed in Japan are discussed.

Keywords. Data highway; distributed control system; data communication; data transmission; process computer control; industrial computer.

INTRODUCTION

The data communication system (Industrial dataway system) is one of the essential function of distributed control system.

In' 70 age, modernized industrial plants have been controlled and managed by using hierarchy computer network system. In here, it was necessitated to introduce a standardized intersubsystem communication in order to increase the degree of compatibility between subsystems supplied by various manufacturers and to combine process control computer system with data processing computer system.

According these requirements, more than 30 kinds of line sharing communication system for high speed data transmission inter computer and inter terminal, were developed and installed in Japanese industries.

In 1975, LSI microprocessors have been utilized to develop new distributed control system. The distributed control system is consisted of process control station, operator console station, data gathering station, other computer apparatus to assemble instrumentation and control system including computer system.

Most important characteristics of the distributed control system are integration of complex and spreaded control and management function in plant or factory.

For well organized system, the data communication is the nervous system in the integrated man-machine system.

In this data communication, both high response and high throughput are required to achive good man-machine communication and data transmission between distributed stations. Recently, many kinds of instrumentation and control stations are implemented to add new function and capability by using LSI microprocessor, and have enough capability to handle data communication and local intelligence.

Under this circumstance, the standardization work has been continued in Committee for Standardization of Industrial Computer Systems sponsored by Japan Electronic Industry Development Association (JEIDA) be concerned with International Purdue Workshop for Standardization of Industrial Computer System and IEC/SC65A/WG6 (Industrial process computer intersubsystem communication).

In Japan, it is recognized the necessity and importance of Industrial Dataway System, and are

PRECEDING PAGE BLANK - NOT FIL

working to prepare the standardized design guideline which describes the design concepts and the basic functions for implementation of industrial process computer intersubsystem communication according Japanese experience.

REVIEW OF CURRENT STATUS

The current line sharing system in the industrial plants are developed and utilized in various type of systems shown in Fig.1 and TABLE 1.

The existing line sharing systems are applicable to support centralized intelligence, hierarchical intelligence, distributed intelligence and/or combinations. So there were many line sharing systems to meet the different grade of needs and requirements.

The basic requirements to characterize each data communication systems are as follows.

1. Volume for data transmission, and required throughput.

2. Quick response in data transmission and priority interuppt handling.

3. Distance of line for communication network.

4. Connected devices to communication network.

5. Intelligence for data communication at each connected devices or stations.

6. Reliability and maintenability.

7. Economics and reducing system costs.

In the first stage of line sharing system, it was applied for process data gathering and monitoring as 1 to N corresponded network structure which was called centralized intelligence. In the integrated computer control system, more high speed data communication system was required and applied for total process supervising and control as n: m corresponded network structure which was called hierarchical intelligency, or computer network. For this purpose, so called data highway systems using few coaxial cables were developed in early of 1970.

After appearance of LSI microprocessor, the distributed control systems are developed and applied to the integrated instrumentation and control system. The systems are designed according two kinds of comcepts. One is assembly of dedicated DDC (Direct Digital Control). The other is assembly of hybrid (amalog/digital) DIO Controllers. Both concepts are supported by using data communication system between each distributed controlliers, operator consoles, and related computer apparatus.

One of the typical implemented industrial dataway system is seen in CENTUM F-BUS which is implemented by YOKOGAWA ELECTRIC WORKS in Japan.

The example of inter-station connection by using F-BUS is shown in Figure-2. It is used to communicate data around the control station by 250K bits per second. It is the branch type network structure, and each station for field process contralling, operator console functions, and computer interface is connected to coaxial cable with passive (transformer) coupler. It can be up to 10 kilometer long, and can be connected through computer interface to virtually any computer having data communication capabilities.
In here, the stations are classified into the primary station and the secondary station at execution phase of communication. The primary station is a station having the initiative of communication. The communication is conducted by a command frame from the primary station and a response frame from the secondary station. It is conducted on a single coaxial cable which conveys a command frame and a response frame one after another.

According these technical trends, industrial dataway system which is capable to support distributed systems for process instrumentation and control must be standardized for future progress of advanced automatic control fields.

DESIGN GUIDELINE FOR IMPLEMENTATION

Herein, the conceptual design guideline of industrial dataway system which is examined by Japanese committee for standardization of industrial computer systems in JEIDA, is preparing one of the proposed intersubsystem communication method in order to inclease the degree of compartibility to various manufactures, thus enabling them to be used together in more complete computer based process measurement and control system.

Type of System

The industrial dataway is digital data communication using bit serial technique over single line sharing system which can be utilized in industrial process computer control and instrumentation. In particular, it shall be useful to support distributed control system.

This data communication is not prepared to take aim at the optimal interface for high speed computing processors with high speed computer interface devices.

This data communication is used for long distance communication (maximum 50 km straight line) at industrial process plant.

It must be specified the device independent functional, electric and mechanical interface requirements which devices and/or station shall meet in order to be interconnected and communicate unambiguously via a system.

Applicable field

The industrial dataway system for on-line real time computer system or distributed control system, is to be used primarily, but not exclusively in the process industries.

From the standpoints of application, the following conditions shall be included.

- To be capable of keeping ultimate correct data communication under industrial process environments
- To provide very high reliability and availability
- To simplify the system assemble, maintenance and training
- To achive economic data communication

Topology and typical configuration

The industrial dataway system is to provide an effective communication link, and to support process data gathering and control, and field expandability. The topology of system is recomended as branch type. Data, control frame, and synchronization signal will be transmitted by using single line. For more reliable system, redundancy like dual or duplex system, or loop communication link shall be applied.

The data transports from one station to another. It is to transmit data directly between any two station and/or subsystem device on the link without necessary involving store and forward at a third station. In data communication link, the master station due to control of a link during a currently transmitting. The slave station dues to receive only during transmission. Master/Slave status of station is changeable, as called multi mastership. But at any given time, master station has responsibility for link control and error recovery during transmission. Address and its related modifier of station and subsystem, shall be specified to be applicable for n:n corresponded communication.

The translator equipment will be available which make it to implement the portions of communication subsystem with common carrier channels. Typical configuration are shown in Fig. 3.

The industrial dataway system will be capable of having minimum 31 stations up to 255 stations. Each station will be connected with the following devices per one stations.

- Process input/output (in case of analog input) ; max. 512 pts
- Process controller (in case of 3 mode controller) ; max. 128 pts
- Terminal device (in case of I/O device subsystem ; max. 16 sets

Transmission mode and characteristics.

The industrial dataway will be optimized for 1 mega bits per second or more and be capable of accommodating the following two classes of data transmission distance to achive broad applicability.
- Remote sites accessible over leased lines with repeater ; Maximum 50 km (straight line)
- Between any two stations ; Maximum 4 km (without repeaters)

This system will be implemented by using bit serial technique and base-band modulation techniques. Frame synchronization will be based on HDLC's definition. Which is standardizen by ISO and also half duplex transmission is good for this purpose, because of high transmission speed.

Asynchronous signals or events so called priority interrupts shall be served within a definite interval that is short time interval to ignore an effect on delay of asynchronous events. The communication system shall be served without delay when the local devices or the terminals have errors or troubles. To satisfy these conditions, priority control down by hardware may not be necessary. Inhere, detection of asynchronous signals or events will be served less than 20 msec or less.

Frame structure

The HDLC's format will be adopted as the frame structure of message for data transmission in this system.

The HDLC's format is follows.

F	DA	C	SA	I	FCS	F

Herein :

F: Flag field (8 bits)
F field designates the start and end of the frame and has the bit structure of
0 1 1 1 1 1 1 0
DA: Destination address (8 bits)
DA field contains the destination address of the station to be received message.
C: Control field (8 bits)
C field defines type of control (ex. command /reply) depending on the bit structure of

this field.
SA: Source address (8 bits)
Basically, SA field contains the source station address of the message. This field is necessary for data transmission on the n:n corresponded communication.
I: Information
I field consists of Header, Control Status, Device address, Text (data) and so on. Length of this field is varied upon amount of data to be transferred.
FCS: Frame check sequence (16 bits)
Each message contains the FCS field for the purpose of detecting transmission error. It is recommendable that checking is based on CRC.

Protocol

The industrial dataway system is structured into five functional levels with respect to data transmission shown in Fig. 4 and TABLE 2 which is based on IEC/SL65A/WG6.

The allocation of specific functional requirements for protocols is defined about physical link, communication link and logical connection as follows.

1. Physical link protocol
The detail requirements of a physical link protocal are dependent on link or line physics and must be transparent and independent of the message bit stream including communication link addresses, control and checking codes, etc.

If message frame delimiters are dependent on line physics, they must be added and subtracted by line coupling unit.
Typical existing standards of this protocol are CCITT V.24 (EIA RS-232), CCITT X.21 and 20mA current loop.

The physical interface of this system is characterized in terms as follows.
- The transmission data rate of 1Mbps or more.
- Bit synchronization.
- Clock extraction (Bit element timing).
- Base-band transmission.
- Electrical isolation from transmission signal line.
- Redundant (Dual or Duplex) data transmission paths should be optional.
- By passing and disconnect capability.

2. Communication link protocol
The communication link protocol is specific interface to the communication subsystem, but is independent from the signalling techniques adopted in the physical link. This protocol is also independent of the characteristics of stations attached to the communication subsystem.

The communication link protocol of this system is organized on the basis of ISO HDLC and characterized in terms as follows.
- n:n (or n:m) communication capability.
- Frame structuring based on the HDLC to transmit unformatted binary data (data transparency).
- Transmission error detection by CRC.
- Error recovery by automatic retransmissions. No-response timer and retransmission counter are required to prevent from unduly delay other messages.
- More than one communication subsystems should be responsible to recover from the disappearance of current master.
- Broadcasting command is required to transmit global messages to all subsystems.
- Initialization capability of communication subsystems.

3. Logical connection protocol
This protocol has to do with the transmitting and the receiving messages. The function must exist in the hardware to be reflected the capability of logical channel read/write interface and be characterized in terms as follows.

- This level shall provide arbitration between the message length capabilities of the communication subsystem and the message length requirements at the application level. Examples are rejection or blocking and deblocking of messages which exceed the message length capabilities of the communication subsystme.
- Establish and release logical data channels.
- Watch Dog Timer to recover from hang-up of logical connection.
- IPL capability for distributed intelligence.

Reliability safety and maintenance

1. Transmission error processing
The industrial dataway system is supporting the fault diagonosing capability.
This capability is classified into two categories. One is included in this system and the other is supported by outside function of this system.

The former includes as the following functions.

- Error control by using CRC checking and automatic retrial functions.
- Sense status of all station.
- Monitoring response time interval.
- Shift the frame synchronizing station or clock unit in case of failure.
- Correction for log of synchronization.
- Data sequence checking.

The latter includes the following functions.

- Code and address error checking.
- Centralized supervisory for all station and communication line operation.
- Test or diagnostics for all stations under on-line operation.

The system is maintaining correct sequencing and integrity of transmitted data through the electrically noisy environments.

2. System failure detection, protection and recovery

According the results of transmission error processing and the system failure detection, the following functions of protection and recovery are required.

- Automatic remote station bypassing by using automatic trouble and error detecting, self loop checking and isolating from active communication link.

- Station bypassing from active communication link without any system disturbance or error by manually.
- Re-organization of communication link (self link configuration)
- Extensive use of selfchecking software should provide reliable supervision to locate and diagnose a potential system error.

3. Smooth and safety operation.

To keep more smooth and safety operation, the following characteristics will be added to the system.

- System expansion and modification under on-line operation of data communication.
- The communication subsystem shall include optional versions which will survive lightning strikes or power faults on the transmission line and will prevent the equipment connected to the subsystem from becoming hazardous when these conditions occur.
- The communication subsystem shall meet the relevant mandatory standards of licensing agencies.

- The communication subsystem shall include optional versions for which intrinsic safety certification can be obtained.
- Easy to make dual stations or dual power supply units.
- Reliable transmission cable way system including dual cabling, protection for disconnection and clamp, and noise protection for transmission signal.
- The physical implementation of the interface within the subsystem should be designed so that the subsystem is isolated and may perform electronic state transitions such as on-line/off-line, power-on/power-off, ready/not ready, busy/not busy, local/remote and so on, without generating transmission errors between other subsystems.

4. Maintenance

The communication subsystem shall be capable of supporting on-line testing and fault diagnosing capability. This might include traffic monitoring and on-line and remote loopback test facilities. The features are not integral functions of the communications subsystem. Also maintenance panel shall be connected to the station for test or diagnostics under off-line mode.

FUTURE PLAN

In Japanese committee for standardization of industrial computer system in JEIDA, study and cooperation for actual implementation of industrial dataway system are settled.

For the work, Japanese committee selected two condidates for further study in IEC/SC65A/WG6. These are CENTUM F-BUS system developed by YOKOGAWA and TOSWAY-1500 developed by TOSHIBA.

Referring these two condidates, we are going to develop the practical design guideline of industrial dataway system with Japanese data communication techniques and operational know-how in the process industries which is one of the basis of Japanese economic growth.

This work will contribute significantly to the evolution of easy assembling, more powerful instrumentation and control system including process control computer system.

Finally, the author wish acknowledge the combined efforts of Japanese committee members and JEIDA peoples. In addition, the author wish to thank Mr. K. Watanabe in Yokogawa, Messrs. A. Uyetani and H. Tanaka in Toshiba and Messrs. T. Saito and A. Furusawa in JEIDA who assisted in supports and reviews of this paper.

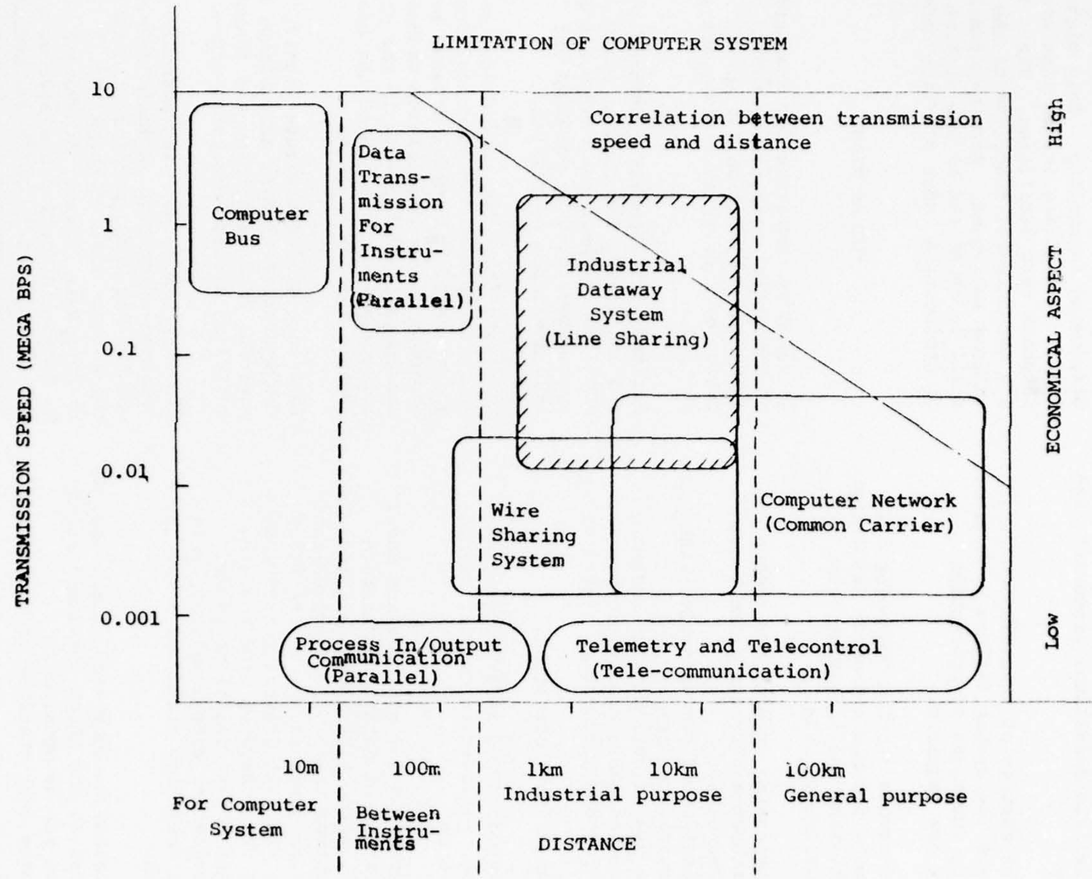


FIGURE 1

STATUS OF DATA TRANSMISSION SYSTEM

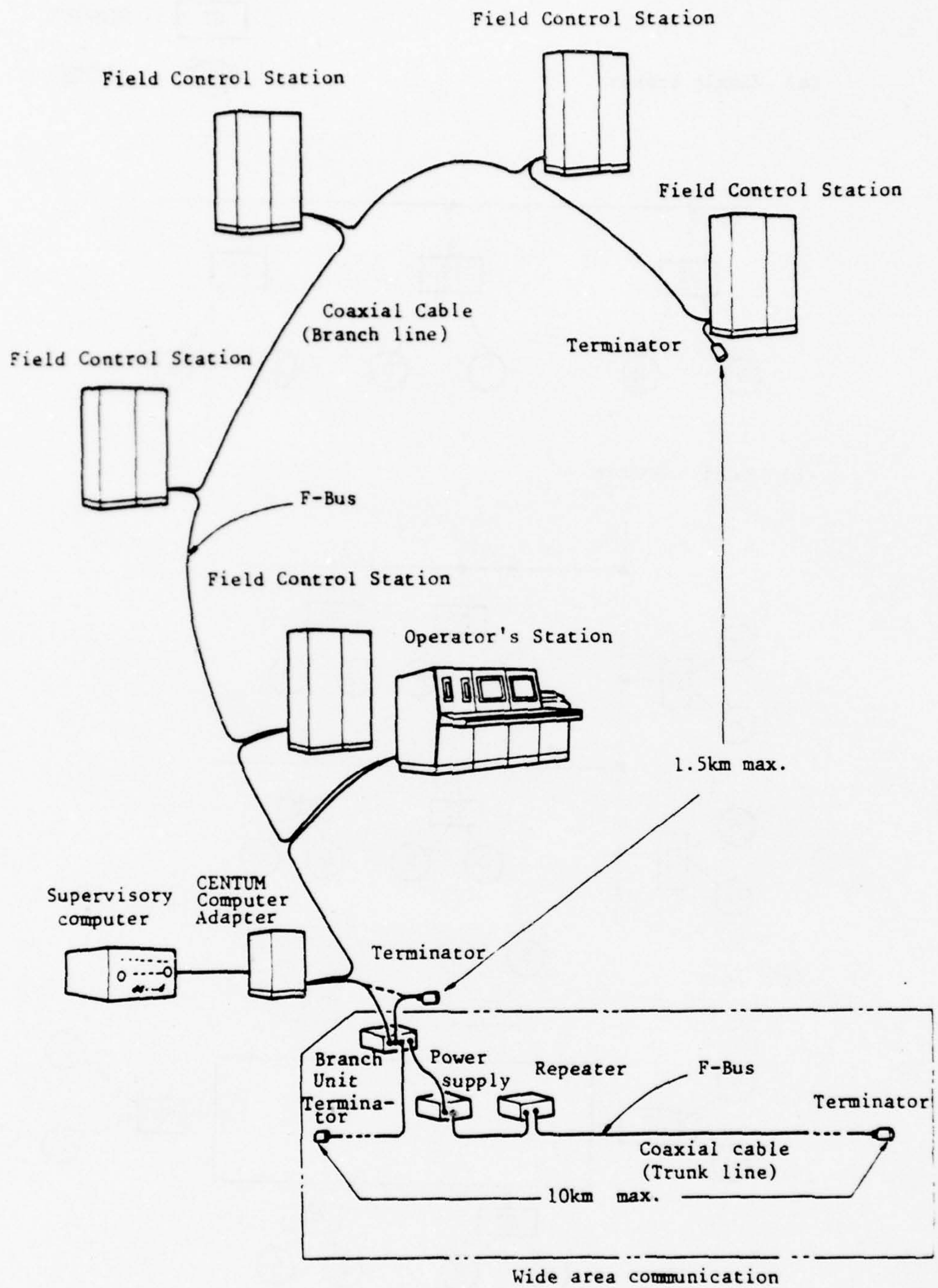


FIGURE 2

EXAMPLE OF INTER-STATION

CONNECTION USING F-BUS

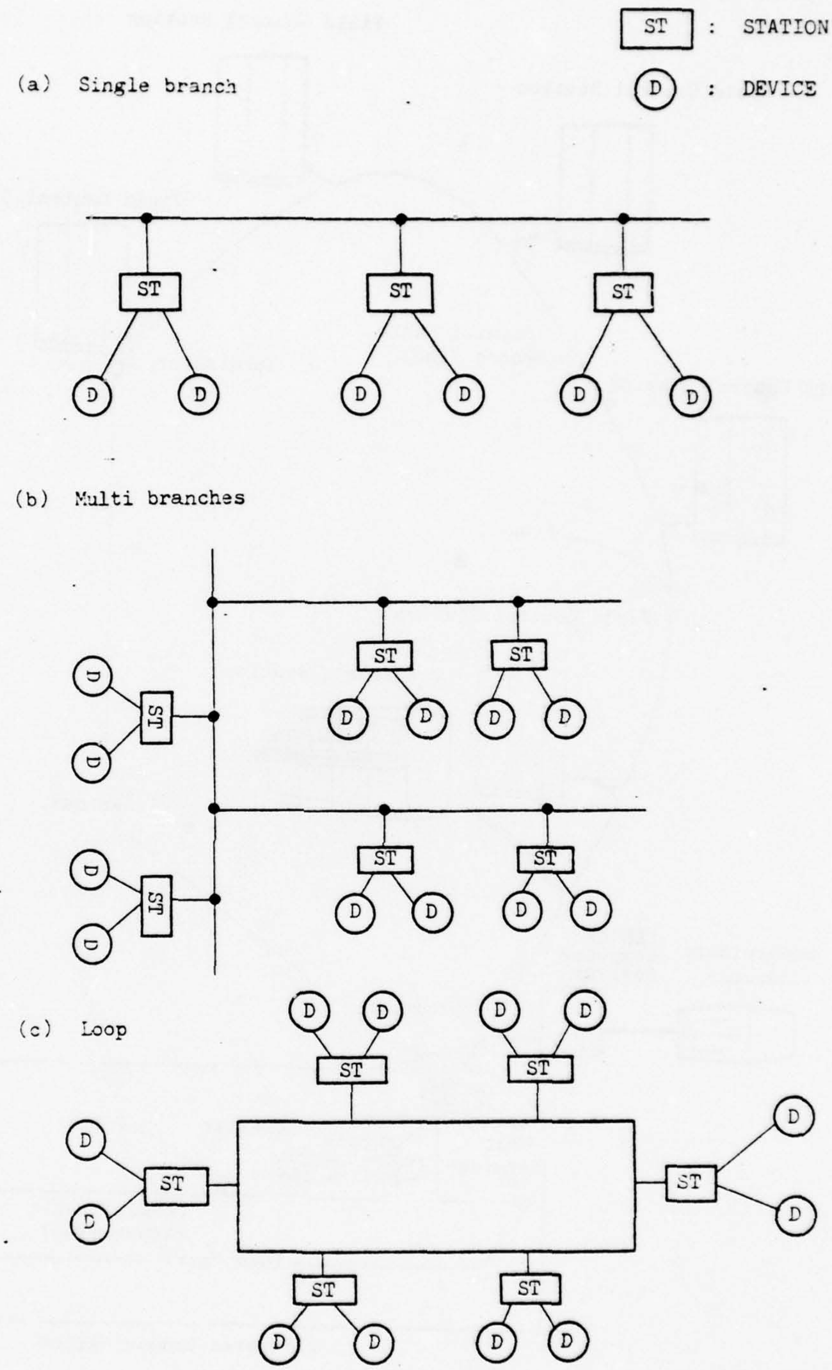


FIGURE 3

TYPICAL CONFIGURATION

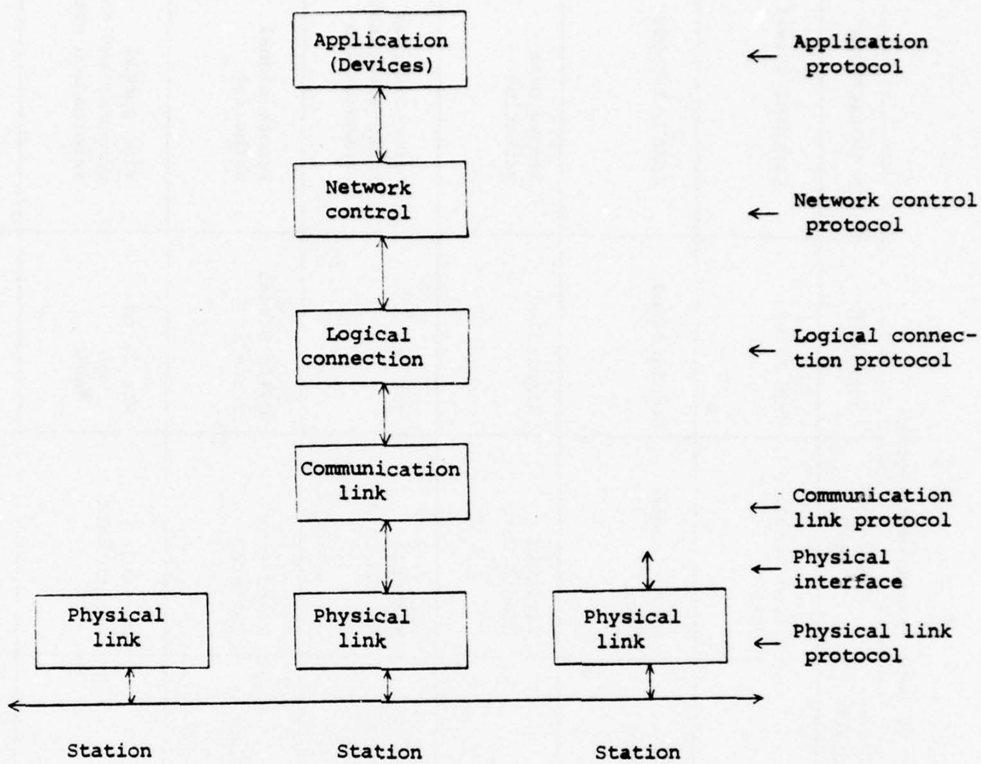


FIGURE 4

SYSTEM STRUCTURE OF THE INDUSTRIAL

DATAWAY SYSTEM

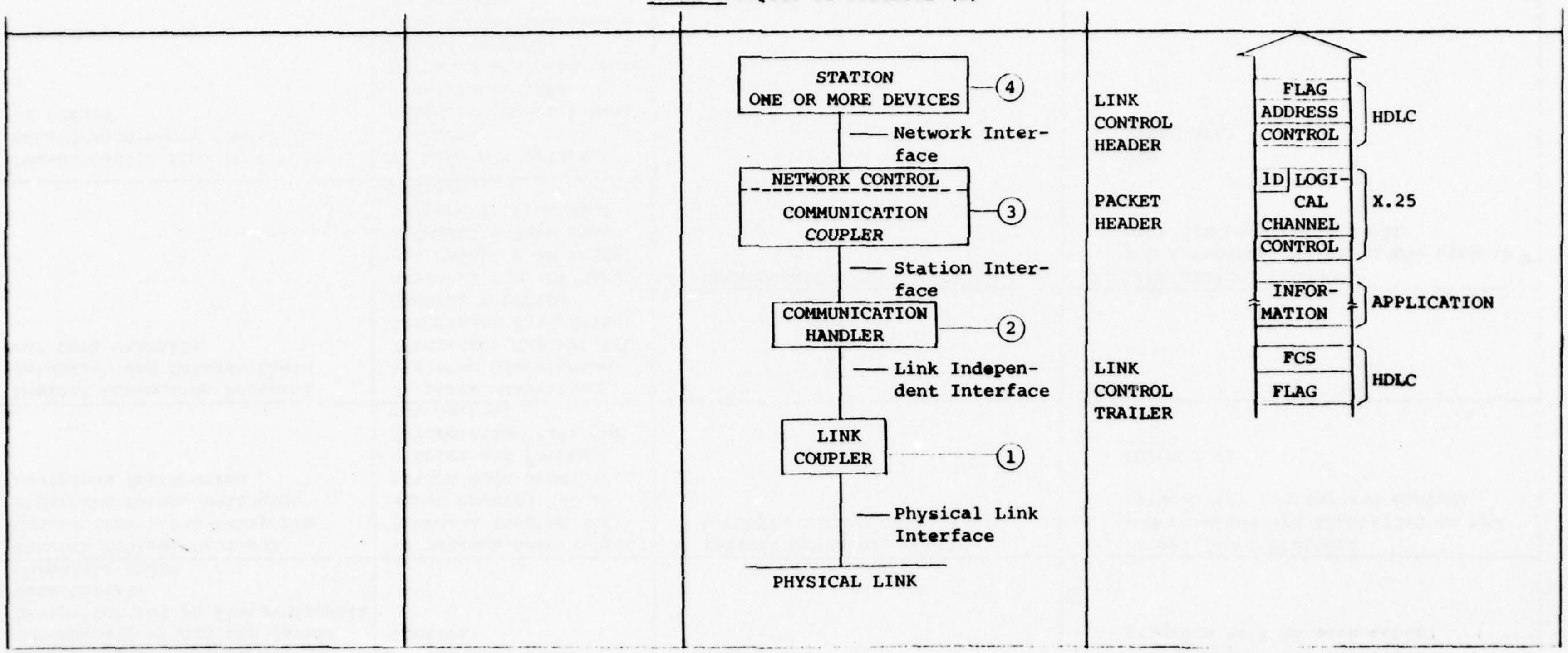
TABLE 1 Classification of Data Transmission System

NO.	CATEGORY	NETWORK STRUCTURE	BASIC DATA TRANSMISSION	BASIC DATA FORMAT	PROTOCOL	MAIN APPLICATION
1	*Computer Bus*	1:1 Correspond or 1:n Correspond		Fixed Length Data	Not fixed	Computer itself
2	Data Transmission for Instrumentation	1:n Correspond or n:m Correspond	Shake Hand Transmission	Fixed Length Data	Standardized	CAMAC, IEC-Bus
3	P1/O Communication	1:1 Correspond	Polling	Variable *Length Data*	Simplified	Remote data *gathering*
4	Industrial Datway System	n:n Correspond	Shake Hand Transmission	Variable Length Data	Not fixed	Inter-computer or inter-subsystem communication
5	Wire Sharing System	1:1 Correspond or 1:n Correspond	Cyclic Transmission	Fixed Bits Pattern	Cylic Serial Transfer	Remote signal gathering
6	Computer Network	1:n Correspond or n:m Correspond	Polling	Variable Length Data	Not fixed or HDLC	Wide spread computer and terminal information gathering
7	Telemetry and Telecontrol or Tele-communication	1:n Correspond or 2:n Correspond	Cyclic	Fixed Length Transmission or Polling	Standardized Data or Byte Oriented Data	Wide spread remote signal data transmission

TABLE 2 Layler of Protocol (A)

ISA SP72(Secretary) 5 (1977-1-14)	Structural Layer of An Interface	IEC TC65A/WG6 Different Levels of Interface & Protocol Activity (1977-1-18)	DATAMATON, Mar., 1976 Layers of Network Access Protocols
5. Application Protocol Device-dependent Information Sub-address within the device Device Control or Status Signals Data Formats Character Codes	5. Application Protocol Device Data Format Control	4. System Protocols and Application Functions	4. Application Level Protocol How terminals and application programs talk to each other.
4. Network Control Protocol Choice from Inter-subsystem Alternate Paths, Switching Decisions for Routing	4. Network Description/ Protocol (System Software) Routing Information Node Service in a Store and Forward Transmission Data Format Control	3. Network Protocol, Routing & Configuration Details	3. Packet Level Protocol How messages are identified to the network for routing and control. CCITT X.25
3. Logical Connection Protocol Establish and Release Logical Data Channels	3. State Transition Skeleton Description (Processor Channel I/O Interface) Read/Write Control Fetching Protocol Seq. of "Are you Ready, I am Ready" & Checking Data Sync. Respect to Processor Storage		
2. Communication Link Protocol Data Transparency, Detection of Errors	2. Line Discipline/ Protocol (Structuring of a Message Frame, Data Blocking and Deblocking, Demand/Response Line Contention, Synchronization Data Checking, Clocking	2. Communication Link Protocol	2. Link Control Protocol How a terminal talks to the network over its physical circuit BSC SDLC, HDLC
1. Physical Link Protocol	1. Physical Interface Characteristics Data Rate, Distance Modulation, Coding Techniques Signal Strength, Coupler.	1. Physical Link Protocol	1. Physical Circuit Protocol How devices are physically connected to the network. EIA RS.232 CCITT V.24, X.21

TABLE 2 Layler of Protocol (B)



APPENDICES J-IV

MICROCOMPUTER WORKING GROUP, PURDUE JAPAN

1. Minutes, First Meeting Microcomputer Working Group, November 18, 1977.
2. Minutes, Second Meeting, January 11, 1978.
3. Minutes, Third Meeting, March 28, 1978
4. *Micro Computer News* - December 1977.
5. *Micro Computer News* - January 1978.

Minutes of the First Meeting of Microcomputer WG IPW-J

1 Date: November 18, 1977
from 2:00 P.M. - 5:00 P.M.

2 Place: Meeting Room of JEIDA

3 Participants:
Yada (ETL)
Togo (Yamatake Honeywill Co., Ltd.)
Kai (Hokushin Electric Co., Ltd.)
Nagagami (Shinko Electric Co., Ltd.)
Otsuka (Oki Electronic Co., Ltd.)
Kobayashi (Omeron Tateishi Electronic Co., Ltd.)
Adachi (Fuji Electric Co., Ltd.)
Kondo (Toshiba Electric Co., Ltd.)
Fujita (Mitsubishi Electric Co., Ltd.)
Uda (Sumitomo Electric Co., Ltd.)
Furusawa (JEIDA)

4 Materials: No. 1-1 Theme of Microcomputer WG IPW-J
No. 1-2 Report on IPW on Microcomputer
No. 1-3 Influence of Microprocessors to Industrial Computer Systems
No. 1-4 INTEL 8273 SDLC Protocol Controller
No. 1-5 Microprocessor controlled Auger Spectrometer

5 Discussion

(1) Inauguration of Microcomputer WG IPW-J

The executive office presented the starting process of Microcomputer WG IPW-J as follows.
From October 3rd through 6th, 1977, International Purdue Workshop was held at Purdue University in U.S.A. At this period, the Steering Committee was held and on this occasion, Microcomputer Committee was established as Ad-Hoc Committee in IPW. And as the Japanese member, Mr. Yada who was attending to this committee, was registerd. Soon after this, the meeting of the committee was held. The main theme of this meeting - "Differences between Minicomputer and Microcomputer" was lively discussed. (ref. material no. 1-1).
On this occasion, it was decided that in future, more lively investigation should be made in every local committee, thus Microcomputer WG IPW-J has been established.

PRECEDING PAGE BLANK - NOT FILMED

(2) A Report on Microcomputer Ad-Hoc Committee of IPW

The activities of IPW and Microcomputer Committee were presented by Mr. Yada (ref. material no.1-1 & 1-2)

(3) Future activity

We have discussed about our course, and the following items were decided.

1).For the present, a chairman is not nominated but rotated.
2) No special obligation is placed.
3) We take up a positive attitude to communication with the foreign countries.
4) The theme of meeting includes every member's interest.

(4) Investigation theme

Each member expressed his opinion as to the study field of his interest as follows.

. Standard Software (Benchmark)
  Software tool of performance appraisal or standardization
. Multi-Micro (Communication between processors)
  Protocol, Distributed system
. Limitation of high level language - from the view of production
. Functional Requirement on Microcomputer for process control
. Reliability, Environment
. Economics of Microcomputer application system
. Development method and tool
. Limitation of Microcomputer application
. Application of Bit Slice computer and its development tool
. Standard Bus
. Debugging tool
. Concept of Microcomputer
. How Software for Microcomputer for control should be.

(5) Theme of next meeting

The theme of next meeting is "Development method and Debugging tool". The following items will be discussed.

1) Development method of Microcomputers

. MDS
. PL/M
. Emulator
. Cross Software/Self Software
. The question of Second source
. Interface Change
. Production of Software
. Counterplan of various machine development
. Development tools of trial manufacture level and Development level

2) Debugging tools

- Logic analyser
- Micro analyser
- Console
- Handi Maintenance
- Software Debugger
- Simulator

6 Next Meeting

Date: January 11, 1978
from 2:00 P. M.

Minutes of the Second Meeting of Microcomputer WG IPW-J

1 Date: January 11, 1978
from 2:00 P.M. - 5:00 P.M.

2 Place: Meeting Room of JEIDA

3 Participants:

Yada (ETL)
Narita (Waseda University)
Kobayashi (Omeron Tateishi Electronic Co., Ltd.)
Yashioka (Yokogawa Electric Co., Ltd.)
Komori (Tohiba Electric Co,, Ltd.)
Yasuda (Hitachi , Ltd.)
Otsuka (Oki Electric Industrial Co., Ltd.)
Togo (Yamatake Honeywell Co., Ltd.)
Hikami (Shinko Electric Co., Ltd.)
Yoshizawa (JEIDA)

4 Materials:

No. 2-1 Microcomputer for Control
No. 2-2 Design and Document Microprocessor Systems for Easy Maintenance
No. 2-3 Development method and Support Software for Microcomputer based devices

5 Discussion:

(1) Confirmation of last meeting report

(2) Summary of last meeting

Mr. Yada explained grouping into six classes as follows which had been investigated in last meeting (ref. material no. 2-1)

Group 1
- Development method for Microcomputers
- Debugging tool

Group 2
- Limitation of high level language
- How Microcomputer Software for process control should be.

Group 3
- Standard Software
- Reliability and Environment of Microcomputer for control
- Concept of Microcomputer

Group 4
- Multi Microprocessor
- Standard Bus

Group 5
- Functional requirement for Microcomputer control
- Economics of Microcomputer application systems
- Application of Bit Slice family and its development tools

Group 6 - Criticism on Microcomputers

(3) Report on Development tools and Debugging tools

First. Mr. Yada made a report on Microcomputer - based devices and its support software of Hitachi. (ref. Material no. 2-3) Next, he explained the state of art of cross software, self software, emulator, logic analyser and so on.
As to standardization of hardware, the current situation was explained that it is pretty well progressed in each place of business but on the whole it is hardly progressed, and the standardization of processor to be used is not well done.
As to language, publication and international standardization of PL/M was proposed.

(4) Future Activity

We have discussed about our course, and the following items were decided.

1) For the present, a chairman is not nominated but rotated as follows.

1 Oki Electric Industrial Co., Ltd.
2 Yamatake Honeywell Co., Ltd.
3 Shinko Electric Co., Ltd.
4 Toshiba Electric Co., Ltd.
5 Yokogawa Electric Co., Ltd.
6 Omeron Tateishi Electronic Co., Ltd.
7 Hokushin Electronic Co., Ltd.
8 Fuji Electric Co., Ltd.
9 Mitsubishi Electric Co., Ltd.

2) We will discuss problems according to the order listed in material no. 2-1.

(5) Next Meeting:

Date: March 28, 1978
from 2:00 P.M.

ELECTROTECHNICAL LABORATORY
TANASHI BRANCH
5-4-1 MUKODAI-MACHI, TANASHI-SHI,
TOKYO, JAPAN
TELEPHONE: 0424 (61) 2141

January 30, 1978

Dear Sir.

The second meeting of Microcomputer Working Group IPW-J was held on January 11, 1978 in Tokyo. The main subject was "Microcomputer Development Technology", and in our discussion, "High Level Programming Language for Microcomputer" was most remarked and a lot of opinions concerning it were offered. We would like to propose you to adopt PL/M of Intel Corp. as Microcomputer Standard Programming Language in IPW.

Now we are expecting to have the comments about Standardization of PL/M. We would be happy if you could give your opinions.

Sincerely yours,

Koji Yada
Koji Yada, Computer Center

CC: Dr. Williams (Purdue University)
Mr. Yoel Keiles (Honeywell)
All Microcomputer Committee Members

Minutes of the Third Meeting of Microcomputer WG IPW-J

1 Date: March 28, 1978
from 2:00 P.M. - 5:00 P.M.

2 Place: Meeting Room of JEIDA

3 Participants:
Yada (ETL)
Narita (Waseda University)
Kida (Tokyo University)
Hashimoto (Mitsubishi Electric Co., Ltd.)
Adachi (Fuji Electric Co., Ltd.)
Kobayashi (Omeron Tateishi Electronic Co., Ltd.)
Hikami (Shinko Electric Co., Ltd.)
Yasuda (Hitach, Ltd.)
Otsuka (Oki Electric Industrial Co., Ltd.)
Yoshioka (Yokogawa Electric Work Ltd.)
Furusawa (JEIDA)

4 Materials:
No. 3-1 A Report on the second meeting of "West Coast Computer Faire"
No. 3-2 An Interim Report on the Application of Microprocessor to Airborne Naval Weapons Systems, by R.D. Hawkins
No. 3-3 NWC Technical Memorandum 3052, by R.D. Hawkins
No. 3-4 An Investigation of Navy Tectical Digital Systems Requirements Microprocessor Applications at NWC, by R.D. Hawkins
No. 3-5 PL/M and its Usage at the Naval Weapons Center, by R.D. Hawkins
No. 3-6 Current Militarized Microprocessors and their Support Software, by R.D. Hawkins
No. 3-7 Theme of Microcomputer WG IPW-J meeting

5 Discussion
(1) Confirmation of last meeting report
(2) A Report on the second meeting of "West Coast Computer Faire"

Mr. Narita made a report on the second meeting of "West Coast Computer Faire" as follows, which was held on March 3rd through 5th, 1978. (ref. material no. 3-1)
Most of all manufacturers except Altair attended the faire. Wholly, the products for education were increasing, and the products for amateur were decreasing. Many of them were for small business with mini floppy and so on, and its cost was $4,000.00 - $5,000.00. One of the inclination was that reliability of peripheral devices which have been developed for amateur have been raised and they have become to be used in the business world.

As to the lecture, he explained the outline, referencing to material no. 3-1.
Many Japanese staffs attended this faire. We are planning to have an opportunity to take their lectures next time.

(3) Management of materials

The following items were decided in regard to foreign materials.

i) Foreign materials must be distributed to all members.
ii) The outline of the deliverated material must be introduced.

Following is a member list that introduce the materials that we have got so far.

1. Material No. 1-3	Hashimoto
2. Material No. 1-4	Hikami
3. Material No. 1-6	Yasuda
4. Material No. 2-2	Kobayashi
5. Material No. 3-2	Kida
6. Material No. 3-3	Adachi
7. Material No. 3-4	Yasuda
8. Material No. 3-5	Otsuka
9. Material No. 3-6	Yoshioka

* next time - 1 - 4

(4) Limitation of high level language

Mr. Otsuka suggested how to proceed discussion of "Limitation of high level language". Then each member expressed his opinion. First an report on feature, production, and object efficiency was made. Its outline is that by using high level language, programming becomes easier and so even beginners can program easy, with the production increasing. However the object efficiency becomes bad. Especially as to PIO peculiar to process control, it can not be used because of its bad efficiency.

(5) As to suggestion of PL/M Standardization

We requested all members of IPW Microcomputer Committee to give their comments on "PL/M International Standardization". Mr. R.D. Hawkins responded to our suggestion pessimistically as follows.

i) LTPL committee of IPW are now investigating of "Long term procedural language"

ii) DOD HOL has already begun the standardization work.

We thanked his comments, and we determined that we would continue studing of "High Level Programming Language for Microcomputer Development System" harder from now on.

6 Next Meeting

Date: May 10, 1978
from 2:00 P. M.

Micro Computer News

(Digest Translated from Japanese)

Vol 2, No. 5, December 1977

English version editor : Koji Yada
Japan Microcomputer Club
Kikaishinko-Kaikan, JEIDA
5-5-8, Shiba-Koen, Minato-ku, Tokyo, Japan

Introduction to Programming

1. Languages for Microcomputers and their Usages

This article points out the merits and demerits of BASIC by showing an example program written in different languages. BASIC is the best language for beginners because its instructions are very comprehensive, errors in programs are detected, and modifications in programs are easy. Also, it permits us to use mathematical processing by a number of functions.

The demerits of BASIC is it takes too much time to execute the programs that includes a lot of repeating instructions. This is because BASIC itself resides in a memory and it needs to be translated. Assembly or Machine languages is convenient which need high speed or the details of I/O control.

2. Text Editor for the 6800 - easy to modify and compile

Text editor is a program which compiles and modify a series of arbitary characters, such as a source program written in Assembly language. In practice it is used to add some lines during a programming and to modify the errors.

Each command used in text editor is explained and the total editor program list is given.

3. Assembler and Assembly Language for Hitachi H68/TR

Assembler is a language processing program which translates a series of numbers in the programs written in machine language into symbols which are easy to be memorized by users.

Assembly language is a program language to write a program by symbolic motation. The smallest unit is called statement. Most of the assemblers read a source program twice: first it makes a label table and second, it translates mnemonic code into operation code.

Hitachi Training Module - H 68/TR Assembler is loaded into ROM with a monitor, and makes object program directly on a memory. It is one-pass type. Therefore you can use assembler without loading an assembler, only have to input a source program once, and execute immediately without loading the object program.

Assembler is convenient to input a source program from key board. Assembler can not fine logic errors of source program because it is a language processing program.

There is an example of program written in machine language, and assembly language. All the statements are explained in details.

4. Step 3 TINY BASIC for the 768-byte BASIC ALTAIR 680b

This TINY BASIC is made to fit into theMITS Cq256 byte x 4 ROM three sockets set on CPU (includes I/O) board of ALTAIR 680 micro-computer. It already has remaind area of about 900 byte for the users in P-ROM monitor (256 byte) and 1K byte RAM.

Memory extention, IO increment, audio cassette interface, process control interface can be built on the board. Byte monitor P-ROM, 1K RAM, serial interface, power supply control switches and case are included in ALTAIR 680 Kit. By adding 3 P-ROM of 768 byte TINY BASIC, TINY BASIC can be soon executed by connecting CRT, TTY at the terminals.

The system variables, functions, operation are explained by a given program and a subprogram.

5. Building MODEM for the communication between the two separate microcomputers

MODEM is an abbreviation for Modulater and Demodulater. It puts data for microcomputers on the waves of voice frequency level and takes data from them.

Tape record, record, telephone can be used to exchange data and it makes easy to handle.

A block diagram of modulater circuit, placement of components of the basic board, circuit diagram of modulater and demodulator, a list of components, are shown.

There are various interfaces with MODEM and the speed of data transmittion can be changed by software.

6. Building the Toshiba Microcomputer Kit TLCS 12A EX-5

This is a new product EX-5 which is built on a single pront basic board 230mm x 180mm. The main part of EX-5 uses 12 bit parallel processing T3190 as CPU. The clock generator circuit resides inside the T3190, therefore the circuit is much simpler, and multiplication and division are included in commands. Toshiba 12 bit microcomputer is used widely in many industry field that it is easy to get peripheral devices. Details of soldering, power supply, basic commands in binary, decimal and hexadecimal numbers, programming are explained.

7. Mini program

Play Cards with microcomputer.

It is a game to find two same numbers in the random numbers. There are 8 kinds of 2 figure numbers in hexadecimal number system.hid behind the 16 keyes. LED displays your count and your partner's count, and the first and the second numbers. 3 seconds after the second key was pushed, two numbers disappear. This game's program is listed.

8. Manual paper tape reader for 100 Yen

Nowadays, data in cassette tape and on records are very popular. But the paper tape is very reliable and it is good ida to keep a very important program in paper tape.

Soldering, I/O port (general-purpose Interface IC, or tristate buffer such as SN 74125, SN 74365) and software are explained.

In order to read data from paper tape only IC sockest is need. But if you add codes, basic board, external 3 IC, it does not cost more than 1000 Yen.

9. Guidance to Microcomputers

This article explains the usage of binary number, especially in relation to its relationship with decimal numbers. MSB is an abbreviation of Most Significant Bit and LSB is an abbreviation of Least Significant Bit. MSB is very important to detect errors. When MSB is wrong the number becomes twice or one half. Also LSB is used to find whether the number is even or odd. In order to change a binary number to a decimal number micro-computer uses division. Decimal numbers and their Binary Coded Decimal are listed.

10. Popular New Products

Examine the functions of Matsushita Microcomputer Kit- Pana Kit KX-33

Pana-kit KX-33 costs ¥39, 800 and it suits best for the training of beginners. The abilities are 1). auto performance of music 2). digital clock 3). timer. It can control microcomputer by light, magnetic water by adding some components. It is very practical and useful in our daily life as various alarms.

Most of the KX-33 kit such as CPU, RAM, Key Switches on board are already finished, therefore only some simple connections are left. It takes less than 2 hours to finish building. There is no need to put long programs in machine language and all the data necessary for each execution can be put later. The main function of this unit is automatic music performance and the limit musical scale is 3 octaves. It can serve as a timer to start music on time and as a buzzer.

11. NIBL Linking with Assembler

NIBL is an abbriviation of National Industrial BASIC language and its indirect operation is one of the most significant function.

Indirect Operator has two types, one is Input and Output ability for parallel words "Peek" and "Poke", and the other is an application to array. How to use this ability is very easy and explained in details.

Link statement is a statement which allows the reference to program which is written in SC/MP in assembler fro NIBL program. When this statement appears in the NIBL program the control of a system immediately is handed over to an assembler program. When an assembler program's processing is over the control again returns to NIBL.

The square root program list is shown.

12. Microcomputer Dictionary

Technical terms; baud, hybrid, LSI MSI, exclusive - OR, Daisy Chain, Unbounding are explained.

13. Questions and Answers

How to make synchronized count down is explained with a count sequence, a truth table of JKFF and control matrixes.

14. News from Japan Microcomputer Club

* Regular Seminars

There are three courses; beginners, intermediate and upper classes.

For beginners, the course includes basic knowledge of microcomputer, and digital circuit; usage of digital IC; How to build a kit; making a power supply; and the basics of software (assembler).

For intermediate course it includes basics of interface, building hexadecimal input key board and LED, Interface of TTY, building journal printer, introduction to BASIC, Building a control panel, Micro OS, Key board, PROM writer, PTR & PTP; approach to application programs and Usage of IC memory.

The upper class course includes building audio cassette recorder, digital cassette recorder, floppy disc, TV display, assembler, Micro OS; Microcomputer programming and bit slice computer; A/D, D/A transformation and music synthesizer.

Microcomputer Circular

1. How to make an Assembler

Backus normal form and Cobol notations are explained. The basic assembler structure is explained with one-pass function, location count, a check of source statement, translation of commands and operands, output of object program, usage of statement table, processing the macro commands evaluation of arithematic equations, post fixed notation, and tree structure.

2. How to write BASIC

Each terms such as statement, program input, command, caluculation mode are explained. A simple program is listed.

3. Interface of Audio Cassette

Command trace circuits, interrupt control, timer interrupt, data format, modelator circuits, and demodelator circuits are explained.

Micro Computer News

(DIGEST TRANSLATED FROM JAPANESE)

Vol. 3, No. 1, January 1978

English version editor : Koji Yada
Japan Microcomputer Club
Kikaishinko-Kaikan, JEIDA
5-5-8, Shiba-Koen, Minato-ku, Tokyo, Japan

1. Key to the selection of microcomputer and peripheral devices

This article gives some good advice to help those who want to buy a microcomputer and peripheral devices. Microcomputers can be devided into two groups for convenience. (1) the family of the 8080 which will be used as a part of circuit -a process controller- when it is ready to perform predetermined functions. The merits of this type are low cost and easy appliance of a support system. (2) the family of the 6800 is suitable when the emphasis is placed upon a extended ability and general purpose rather than dedicated. This group may become a base for the execution of a program development and other tasks.

Some single board microcomputer are introduced with the detailed descriptions; SDK-85 (Intel Japan), TK-80 and TK-80E (NEC), MP-80 (Logic Systems International), PROTO 80 (MT-Corporation) 6800 D II A (Motoroller), Lkit-8 (Fujitsu), and many others are included.

Various models of peripherals such as key boards, encoder, TV display, printer, digital cassette tape memory, paper tape reader, card reader, and power supply are presented. This article suggests increasing use of 16K dynamic RAM as a main memory in consideration of peripheral device's cost.

2. *How to apply a microprocessor*

When you plan to build a device which requires a certain degree of complex processing , it is not easy to decide which logic to use, hardware, or software. This article shows how easy to implement a microprocessor and also gives a good example. A small response analyzer (RA), also is called Response Wave Analyzer, is a simple data calculating device. It is necessary to discuss thoroughly before the final specification is made. In a small development like this, the agreement in this step of a development greatly influences upon the duration of the development and its quality. The design with hard logic is explained first. When the basic boards for training

and OEM which are available from many manufactures are used, the total cost becomes much cheaper than buying IC chips and building by oneself, except for very special specifications MP80, a microcomputer board with a resided monitor is used. This can also perform its debugging on this board. A program list and how to write in the program are described in detail

3. Concept and building of a display of Morse Signal Automatic Decoder

This article is the second series to introduce programs which is oriented to a practical use of a microprocessor as Morse Signal Decoder. It explains how the signals are decoded by a microcomputer. Generally TV sets and small printers are used to display characters by a computer.
But both of them are expensive. So 35 LED are used as a 7X5 LED Matrix decoder. This device's feature is its low cost. Flow charts of decoder algorithm are explained.

4. Building up a simple TV display

By picking up a slow output signals, we can see the singals. Detailed diagrams of circuits are listed. How to modulate and how to use it for other application are explained.

5. Special Report - The future Robot Society with microcomputers

This report summarizes the 7th International Symposium of Industrial Robot (ISIR) which was held in Tokyo in October 1977. The theme was discussed with about 80 representains from 16 countries. This symposium was divided into 4 parts.

1. Development of hand mechanism
2. Development of various working robots
3. Use of microcomputers
4. Large Scale Projects

As far as Industrial Robots are concerned, the economic and social evaluation, controlling methods, softwares of controll, mechanical hand, standardization, new mechanism and its applications, and artificial intelligence robot are represented. This reports points out one of the features which will be considered for future developments, that is, its flexibility of material, structure, and functions.

6. Basics for Mini-Program

This is the second series of explaining how to open square roots. This is the fastest method which was found by listing the numbers of adding odd numbers and its sums. A program list are explained thoroughly..

7. 12K byte ROM/RAM board

A new 12K byte ROM/RAM board with multi addressing is introduced from Adteck System Science. ADB-001 is ROM/RAM basic board compatible for 8080 and 6800 Chip selection, Output lines for TVD-02, Output lines for TK-80, combinational use of ROM/RAM are explained.

8. Common sence about microcomputers

Basics of microcomputer functions, merits and demerits, techniques, various models are explained.

9. Microcomputer Guidance 3

This article points out why binary numbers are used in microcomputers. The principal merits are reliability because it uses only 1 and 0, and number of states for circuits are much less than decimal numbers.

10. Introduction of New Products

EX-12/5 (Toshiba), TK-80 BS system and various Interface for L Kit (Panafacom)

11. Microcomputer Dictionary

Technical terms such as threshold, rise time and full time, MSD, LSD, idle time, POS, acknowledge signals are explained.

News from Japan Microcomputer Club

The second Microcomputer Show '78 will be held in Tokyo on May 18. There will be four parts.

1. microcomputers
2. microprocessors
3. application device for microcomputers and microprocessors
4. manuals and books

APPENDICES A-VIII

TC-1

REAL-TIME INDUSTRIAL FORTRAN COMMITTEE

PURDUE AMERICAS

1. Caro, Richard H., An Update on the Relationship Between Tasking and File Handling, S61.2 and S61.3, May 22, 1978.

2. _Industrial Computer System FORTRAN Procedures for the Management of Independent Interrelated Tasks_, Draft ISA S61.3, March 30, 1978.

International Purdue Workshop
Industrial Computer Systems

PURDUE LABORATORY FOR
APPLIED INDUSTRIAL CONTROL
102 Michael Golden
Purdue University
West Lafayette, Indiana 47907, USA
317/494-8425

JUN - 6 1978

Please reply to: Mr. Richard Caro
Dept. 330
The Foxboro Company
Foxboro, MA 02035
USA
TELEX 927602

May 22, 1978

Members of TCIA/ISA S61:

The attached working paper is an attempt on my part to resolve the differences between the ISA documents S61.2 - 1978 and "Industrial Real-Time FORTRAN" as proposed by TCIE, Purdue Europe, in the field of File Handling. It also re-casts the file handling aspects more in the light of tasking. Finally, I also attempted to show the effect of FORTRAN 77 on S61.2. All of this was not possible at the time S61.2 was being written because we didn't know enough. Now that we are all smarter, it seems appropriate to review these items for future revision considerations.

R. H. Caro

R. H. Caro, Dept. 330

ljk

Attachment

PRECEDING PAGE BLANK - NOT FILMED

AN UPDATE ON THE RELATIONSHIP BETWEEN TASKING AND FILE HANDLING, S61.2 AND S61.3

By R. H. Caro

Industrial real-time computing has a set of requirements which are both different from and yet similiar to other fields of computing. Direct, random inputs from industrial processes make impossible the ability to organize a series of programs in such a way as to be able to predict the exact order of execution. While such situations often exist in data processing applications, they are usually confined to the I/O operations designed into the operating system. Industrial real-time programming requires the user to be able to directly address these problems in his program design. Thus some of the techniques usually required to write operating systems must be made available to the user of industrial real-time programming.

The model processor for the industrial real-time environment must be considered to actually support multiple concurrent executions of programs. Each processor must have access to common memory and the same I/O devices. Thus it is part of the model that there will often be actual contention for the same device or data item. This model has been referred to as a virtual processor which may indeed exist, as a real processor with modern technology, or may be mapped onto a processor with only one actual program in execution at a time (others in suspended state).

There are four general topics which industrial real-time programming languages must address:

1. Scheduling of program execution
2. Allocation of critical resources
3. Sharing of data
4. Input/Output with binary or bit-string devices.

Scheduling must be based on time or event occurance or in some cases, both. Allocation of critical resources require a method which excludes possible contenders during the time when the resource is being used. Sharing of data must be done in a way which is cooperative yet provides sufficient safeguards to protect data integrity. The I/O devices usually associated with industrial real-time applications, tend to be primative devices of a wide variety such that their data interfaces are usually expressed as a simple binary state or as a string or strings of binary bits.

The ISA through its S61 committee has attempted to address all of these issues through the medium of standards development. The I/O problem is addressed in ANS/ISA S61.1 - 1976, Industrial Computer System FORTRAN Procedures for Executive Functions, Process Input/Output and Bit Manipulation. The control of the only standardized method in FORTRAN of sharing data, the file, is addressed in ISA/S61.2 - 1978, Industrial Computer System FORTRAN Procedures for File Acess and the Control of File Contention. Finally, the remainder of the issues are addressed in the draft standard ISA/S61.3, Industrial Computer System FORTRAN Procedures for the Management of Independent Interrelated Tasks.

The following discussion of file handling is an update to ISA S61.2 in the light of the approval of ANS X3.9 - 1978, FORTRAN 77 and the comments of the Purdue Europe FORTRAN Committee. Also, it shall be recognized that a very advanced method of task synchronization is called "message passing", which could be implemented using exactly the mechanism discussed here.

FILE HANDLING

The intent of ISA/S61.2 is specifically to enable the executive program to obtain enough information about the intended use of the particular form of shared data known as the file so as to control the several possible contenders for that resource. Consideration must be given as to how the program at hand intends to use the file as well as any knowledge as to the degree to which it might be possible to allow simultaneous access to that same file. The file system itself is not the object of the standard. As such, the series of FORTRAN subroutines are designed to allow control of the shared resources called "files" in a FORTRAN environment.

Unfortunately, FORTRAN, X3.9 - 1966 did not adequately provide for the kind of files most often found in the real-time industrial control environment; namely, random (or direct) access named files. Therefore, additional FORTRAN subroutines had to be provided to substitute for these file system features. These were the following:

CFILW	(create a file)
DFILW	(delete a file)
OPENW	(open a file)
CLOSEW	(close a file)
RDRW	(read a direct access file)
WRTRW	(write a direct access file)

Now that FORTRAN, X3.9 - 1978 (FORTRAN 77) has been approved, it is time to re-examine the place and need for ISA/S61.2. FORTRAN 77 provides for all of the above file operations but without the necessary access controls. If ISA/S61.2 were to be revised in this light, it should be to add subroutines which would provide the necessary access controls prior to the actual "OPENING" of the file by normal FORTRAN 77 code. Alternatively, sufficient information could be expressed in the subroutine call itself to accomplish both the "OPEN" function and obtaining the devised access controls atomically within the subroutine. Either method would provide sufficient control. The first method would be much simpler, but requires that the user enforce his own dicipline to be sure that proper opening and closing access controls are indeed used. The second method tends to assure proper use since normal FORTRAN code would not be used, but the syntax of OPEN and CLOSE in FORTRAN 77 would require an exceedingly complex expression of the arguments to such subroutines.

To reach a reasonable compromize, a short set of different subroutines could be provided to reflect the "usual or most frequent" requirements. Then, if other functions were required, a two step more general procedure could be used. The proposal consistent with FORTRAN 77 would be as follows:

CFILW	(create a new file and open it)
OPENW	(open an existing file)
CLOSEW	(close a file but KEEP it)
DFILW	(close a file and delete itfrom the system)

The file access modes are the expression of the user as to the intended use of the file so that the executive program can assign file access rights to the set of simultaneous contenders. It is necessary to understand that certain files may be assigned properties at the time of their creation which may not allow certain modes of access. The executive program must resolve such issues. The original intent of ISA/S61.2 was to allow a maximum degree of sharing when simultaneous contenders are present. S61.2 allowed four forms of access mode as follows:

·Shared - unconstrained access, maximum sharing

·Exclusive - totally constrained access, no sharing

·Read-only - caller only reads, others may read or write.

·Protected Read - caller only reads, others may only read.

Upon close eximation of these functions, it is difficult to distinguish a useful difference between the Shared and the Read-only modes.

To better understand the necessity for controlling access to files, we must return to the concept of a file as the only controllable means of accessing shared data. With FORTRAN 77 allowing files to be "internal" storage as well as external, inefficiency is no longer a concern. Therefore, it is proposed that file access modes be simplified to be similiar to expressions of critical region control as follows:

.Locked - file opened to only the calling program. Includes the previous Exclusive mode.

·Unlocked - file opened to all requestors. Includes both the Shared and Read-only modes.

·Protected - file opened to all requestors who also open in the Protected mode. Includes the previous Protected Read mode.

New requests to open files already opened under the above modes would be resolved as in Figure 1. Requests to open a closed file would always be accepted, unless processor dependent properties of the file prevented such a mode of opening; for example, a "read-only" file would be proper to open in any mode, but a "write-only" file would be improper to open in Protected mode.

	File Already Opened in Mode		
Request to Open in Mode	Unlocked	Protected	Locked
Unlocked	Yes	No	No
Protected	No	Yes	No
Locked	No	No	No

Figure 1
Resolution of File Opening Requests

Finally, a means to modify the access modes must be supplied to change the mode without closing the file (which might damage integrity of the file with concurrent program execution). This subroutine would be MODAPW, similiar to the contents of ISA/S61.2.

With respect to the proposals of the Purdue Europe FORTRAN Committee, Industrial Real-Time FORTRAN, the new file modes would include their old modes as follows:

- Locked - includes "Exclusive" (and certain cases of "Write-Only").
- Unlocked - includes "Read/Write", "Unprotected Read" and "Write-Only".
- Protected - Includes "Protected Read".

Some doubt exists as to the necessity of a "Write-Only" access mode. Write-Only is much more of a description of a file access privilege which is assigned to the file itself, and is therefore outside the scope of S61.2. If it is desired to allow concurrent writers to such a file (as for an alarm message file), then it would be opened "Unlocked". If message integrity must be preserved, then it should be opened "Locked."

Draft ISA/S61.3

INDUSTRIAL COMPUTER SYSTEM FORTRAN PROCEDURES FOR THE MANAGEMENT OF INDEPENDENT INTERRELATED TASKS

March 30, 1978

Note: This copy of ISA draft proposed standard S61.3 is being circulated for comment only. It is a working paper of ISA/S61 committee and is subject to change without prior notice. Please address all comments to the chairman:

Dr. Matthew R. Gordon-Clark

Scott Paper co.

Scott Plaza III

Philadelphia, pa 19113

USA

PREFACE

This Preface, all footnotes, and all Appendices ae included for informational purposes and are not part of Standard ISA-S61.3.

This Standard has been prepared as a part of the service of the Instrument Society of America toward a goal of uniformity in the field of instrumentation. To be of real value, this document should not be static but should be subjected to periodic review. Toward this end, the Society welcomes all comments and criticisms and asks that they be addressed to the Standards and Practices Board Secretary, Instrument Society of America, 400 Stanwix Street, Pittsburgh, Pennsylvania 15222.

The ISA Standards Committee on Industrial Computer System FORTRAN Procedures SP61, operates within the ISA Standards and Practices Department, A. P. McCauley, Vice President. The SP61 Committee also is a working group of FORTRAN Committee of the International Purdue Workshop on Industrial Computer Systems which provides the impetus for the development of this and related standards. The FORTRAN Committee is chaired by M. R. Gordon-Clark; General Chairman of the International Purdue Workshop is Dr. T. J. Williams.

Committee SP61, Industrial Computer System FORTRAN Procedures was established in June 1971 as a working group of the FORTRAN Committee of the Purdue Workshop on Standardization of Industrial Computer Languages (later reorganized and renamed the International Purdue Workshop on Industrial Computer Systems). As a result of their activity, ISA Standard S61.1-1972, Industrial Computer System FORTRAN Procedures for Executive Functions and Process Input Output, was published by the ISA in 1972. In 1974, extensive use of S61.1-1972 and discussions with ANSI X3J3 FORTRAN Committee indicated a need for revision and the standard was revised as S61.1-1976.

In 1977 ISA S61.1-1976 became an American National Standard ANSI/ISA S61.1-1976 and was submitted to the International Standards Organization working group ISO/TC97/SC5/WG1 for consideration as an international standard. The standard ISA S61.2-1977, Industrial Computer System FORTRAN Procedures for File Access and the Control of File Contention, was also developed by the SP61 Committee in conjunction with the FORTRAN Committee of the International Purdue Workshop in particular the American and European branches of the workshop.

The development of S61.3 began in 1972 and the standard is the result of many different inputs which were coodinated through the International Purdue Workshop.

Many different people contributed to the deveopment of this standard, but the ideas and comments of the FORTRAN Committee of the European branch of the International Purdue Workshop, under the chairmanship of Professor G. Heller were especially important. The standard was prepared by a working group of the International Purdue Workshop FORTRAN Committee which constituted the SP61 committee listed below.

M. R. Gordon-Clark, Chairman	Scott Paper Company
A. Arthur	IBM Corporation
F. E. Bearden	Modular Computer Systems
R. H. Caro	The Foxboro Company
L. M. Cartright	Inland Steel Company
W. Van Diehl	Hewlett-Packard Company
M. N. Hands	Digital Equipment Corporation
C. C. Haskell	Union Carbide Company
T. L. Luekens	Johnson Service Company
O. Petersen	Norwegian Institute of Technology
R. E. Signor	General Electric Company
R. W. Swearingen	Virtual Systems Inc.
D. W. Zobrist	ELDEC

1.0 SCOPE

This standard presents external procedure references for use in industrial computer control systems. These external procedure references provide means for task intercommunication and cooperation through the orderly management of independent task execution. The tasks are expected to be running in a multi-tasking or a multi-processing environment. These procedures usually will pass the information necessary for this orderly management to an executive routine which will ensure the correct operation of industrial computer control system. The method for ensuring orderly management is left to the processor.

The procedure references in this standard are intended to provide methods by which the task can inform the processor of the relationships between the task and other independent tasks under the control of the same processor. The standard provides the means to integrate independent tasks into a system of cooperating tasks when used in conjunction with sound program design but the implementation of this standard is no assurance that problems of non-cooperation will not arise.

1.1 DEFINITIONS

VIRTUAL PROCESSOR

A virtual processor is an environment in which an executable program can run and in which all the resources needed for the execution of the executable program are always available. A particular implementation serves to map a set of virtual processors on to the actual processor. This mapping may be controlled by an executive routine and is processor dependent. The virtual processor exists only when the set of pending executions is not empty, or when an executable program is in either the state Voluntary Suspend on the state Running.

CYCLIC EXECUTION

Cyclic execution is a set of executions of an executable program by its virtual processor.

FORTRAN

The computing language defined as full FORTRAN ISO R1539-1972 which is the same as American National Standard FORTRAN ANSI X3.9-1966.

MULTI-PROGRAMMING*

A mode of operation that provides for the interleaved execution of two or more computer programs by a single processor.

MULTI-PROCESSING*

A mode of operation that provides for parallel processing by two or more processors of a multi-processor.

MULTI-TASKING*

A multiple operation that provides for the concurrent performance or interleaved execution of two or more tasks.

PROCESSOR*

In a computer a functional unit that interprets and executes instructions.

TASK

The execution of an executable program. The ISO/DIS 2382/X definition of a task is more general, "A basic unit of work to be accomplished by a computer". In this standard the basic unit of work is restricted to an executable program.

EXECUTABLE_PROGRAM

A program in a form suitable for execution.

INDEPENDENT_TASK

A task whose existence at any time is independent of the existence of any other task.

DEPENDENT_TASK

A task whose existence is dependent on the existence of another task.

CYCLIC_EXECUTION_OVERRUN

Cyclic execution overrun occurs when a task is a set of cyclic executions of a program and when the start of the execution of a program occurs before the completion of the previous execution of the program.

EXECUTIVE_PROGRAM

Software under control of the processor which is responsible for the orderly scheduling of system resources allocating them as requested by tasks or as previously directed based upon event occurrence.

----

* Definitions taken from ISO/DIS 2382/X

1.2 Background Information

In a computing system individual tasks may have various relationships.

--Tasks are independent.

--Tasks are independent but interrelated.

--Tasks are dependent.

Industrial systems usually consist of independent but interrelated tasks and in these systems the tasks can have various requirements such as:

--Tasks can be initiated either at a specified time or by an event and such initiation can be carried out repetitively.

--Tasks can be initiated to be complete by a specified time. This is frequently called deadline scheduling.

--Tasks can be synchronized to clock time or with process events.

--Initiation can be deferred depending on time or event.

--Tasks can be suspended voluntarily or involuntarily.

--Tasks can be terminated through normal completion or through unforseen exception conditions.

--Tasks may share information.

--Tasks may send messages to other tasks.

--Tasks can terminate other tasks.

--Tasks can suspend other tasks.

--Tasks can be resumed after suspension.

--Tasks may set, clear or test event indicators.

--Tasks may obtain unique control of a shared resource.

1.3 Task Features Included In This Standard

In industrial computer systems, tasks are independent but interrelated. This standard does not address all the areas of independent task interrelationships but is concerned with those features that most commonly arise in industrial computer systems.

Table I shows those features covered by the standard and those excluded; however, the excluded features may affect the result of a request for cooperation between independent tasks and such effects are processor-dependent and outside the scope of this standard.

TABLE I

INCLUDED

Independent but interrelated tasks
Initiation at a given time, upon event, after a period or repetitively
Suspension of a task by itself
Resumption of a task by another task
Resumption of a task by an event, at a given time or after a period
Termination by a task of another task or itself
Setting/clearing/testing of event conditions

EXCLUDED

Dependent tasks
Processor-dependent exception conditions such as overflow/underflow
Task suspension by another task
Suspensions caused by the processor
Suspensions caused by the executive routine
All methods of sharing variables or arrays
All methods of sharing files
Sending messages between tasks
Termination of tasks based on events in the executive routine
All functions of the executive not involving task management
Deadline scheduling of tasks
Boolean operations on events

2. STATES OF A TASK

A task can exist in four states which define the operation of the task. All procedure references concerned with task interrelationships involve the change of a task from one state to another state.

The states of a task are defined in terms of a task's virtual processor. A virtual processor is defined in Section 1.1. A virtual processor has all of its resources always available for the execution of the task. The executive performs the necessary mapping of a set of virtual processors onto the actual processors. The mechanism used for this mapping is processor dependent. Included in the resources necessary for task execution are: memory space, central processor(s) time, and general computer peripheral availability. The mapping of the virtual processors onto the real processor frequently implies that the time for a task to execute to completion the real processor is greater than the time for the same task to execute to completion in its virtual processor. Thus the mapping of the virtual processors onto the real processor is a measure of system loading which is clearly both system design dependent and processor dependent.

Each separate initiation of a task creates a new virtual processor. Each such virtual processor is identifiable to the executive program and the mechanism for identification is processor dependent. A cyclic or repetitive initiation of a task by a single procedure reference has only one virtual processor for all the repetitive executions of the task and is identifiable as a single entity to the executive program.

A diagram of the states is shown in Figure 1.

The definition of task interrelationships in terms of virtual processors does not imply that a particular implementation must include the concept of a virtual processor or require a virtual storage executive routine, but the particular implementation must ensure that the result of an execution of a reference to the subroutine procedures given in Section 4, follow the definitions in that section.

A task can change the state of any other task except that a task can only be changed from Running to Voluntary Suspend by itself.

2.1 Task State Definitions

The four states of a task are Non-existent, Pending, Running, and Voluntary Suspension defined as follows:

2.1.1 Non-existent. The task's virtual processor does not exist and the real processor is unaware of the existence of the task. However, the executable program, whose execution would become the task, can and usually does exist and the executable program's existence is usually known to the real processor.

2.1.2 Pending. The task's virtual processor exists and the task is awaiting the occurrence of an event to begin execution at which occurrence the task will move to the state Running. The real processor may or may not have loaded the task into its memory but the task is known and identifiable to the real processor. The mechanisms for this identification are processor dependent.

2.1.3 Running. The task's virtual processor exists and the task is running in its virtual processor. At any time the task may or may not be running in the real processor depending on the mapping procedure of the virtual processors onto the real processor. The mechanisms for this mapping are processor dependent.

2.1.4 Voluntary Suspension. The task's virtual processor exists and task is suspended in its virtual processor awaiting the occurrence of an event to resume execution at which occurrence the task will move to the state Running in its virtual processor. The time at which the task resumes executing in the real processor is processor dependent as it is affected by the mapping of the virtual processor onto the real processor.

```
        +-----------+      A      +-----------+
        | VOLUNTARY |<------------|           |
        |  SUSPEND  |             |  RUNNING  |---------+
        |           |------------>|           |         |
        +-----------+      B      +-----------+         |
              |                      ^   ^              |
              |                      |   |              |
              |            +---------+   | F            | E
              | E          |             |              |
              |            |          +-----------+     |
              |            |          |           |     |
              |          C |          |  PENDING  |     |
              |            |          |           |     |
              |            |          +-----------+     |
              |            |            |   ^           |
              |            |          G |   | D         |
              |            |            |   |           |
--------------------------------------------------------------
NON-EXISTENT  |            |            |   |           |
              V            |            V   |           V
```

Figure 1

TASK STATE DIAGRAM

A) DELAY, WAIT, WAITE
B) occurrence of the event or completion of the time period
C) SKED, START, TRNON for immediate execution or CON where
   the eventmark is already ON
D) SKED, START, TRNON, CON, CYCLE
E) ABORT, STOP
F) occurrence of the event or time
G) DSKED, DCON, DCYCLE

3. EVENTMARKS

In the management of independent interrelated tasks, there are numerous requirements to key actions to the occurrence of events. The mechanism chosen to enable such action is the eventmark.

An event is noted by incrementing a specific eventmark by one (1) and is called "setting" the eventmark. The eventmark is decremented by one (1) by direct program control or by the execution of a task which was keyed to the occurrence of the event noted by the eventmark. This action is called "clearing" the eventmark. Eventmarks can only be changed by the procedure references defined in this standard and by the executive routines with which it services the events.

If an eventmark is defined as zero (0) in a procedure reference, no action will be taken and the executive program will ignore any actions dependent on that eventmark.

3.1 Eventmark Handling

Eventmarks are properties of the processor which exist as zero or positive integer values where zero is considered as OFF and any positive value is considered ON. Eventmarks are cumulative, in that they will count the number of "sets" thus requiring an equal number of "clears" before the condition will change. If an eventmark is OFF, the eventmark will not count the number of "clears" so that a single "set" will always establish an ON condition.

Eventmarks can be changed by two different types of mechanisms:

1. By the execution of reference to the subroutine SETEM, CLREM, and PRESEM; or

2. by the executive program as it services the programs associated with the eventmarks.

The executive program must automatically maintain eventmarks in such a way as to denote the present state of the real-time associated event. Thus, the action of the executive program in servicing an eventmark ON condition will be to either schedule a PENDING task and/or to move a VOLUNTARY SUSPENDed task to the state RUNNING, if such tasks indeed exist in such states. To denote such actions, the executive program will decrement the eventmark. If the action of decrementing the eventmark does not cause the eventmark to become OFF, further service of the eventmark must occur. If no tasks are associated with the particular eventmark, no decrementing will occur.

3.2 Setting an Eventmark to the ON Condition, SETEM

Execution of a reference to the subroutine SETEM shall cause the specified eventmark to be set to the ON condition. If the eventmark was already ON, the accumulation of such sets shall be incremented. The form of the CALL is as follows:

CALL SETEM (e,m)

where:

e specifies the desired eventmark; an integer expression

m is set on return to the calling program to indicate the disposition of the request as follows:

The value must be 1 or greater

1 - request accepted

2 or greater - request rejected

This argument shall be an integer variable or integer array element.

3.3 Clearing an Eventmark, CLREM

Execution of a reference to the subroutine CLREM shall decrement by one the count of sets to the specified eventmark. If the eventmark was already OFF, there will be no action. The form of the CALL is as follows:

CALL CLREM (e,m)

where:

e specifies the desired eventmark; an integer expression

m is set on return to the calling program to indicate the disposition of the request as follows:

The value must be 1 or greater

1 - request accepted

2 or greater - request rejected

This argument shall be an integer variable or integer array element.

3.4 Testing the Eventmark Condition, TESTEM

Execution of a reference to the function TESTEM shall return a logical TRUE value if the specified eventmark was ON and a logical FALSE value if the eventmark was OFF. If the eventmark is unknown to the processor, a logical FALSE value will be returned. The condition of the eventmark shall not be affected. The form of this function reference shall be as follows:

TESTEM (e)

where:

e specifies the eventmark, an integer expression

3.5 Presetting an Eventmark Count, PRESEM

Execution of a reference to the subroutine PRESEM will assign a declared value to the specified eventmark. The form of the call is as follows:

CALL PRESEM (e, v, m)

where:

e specifies the eventmark; an integer expression

v specifies the desired preset value of the eventmark; an integer expression which must be zero or positive in value.

m = 1 accepted

= 2 queue of waiting / scheduled program, not empty

$\geq$ 3 other error

m is set on return to the calling program to indicate the disposition of the request as follows:

The value must be 1 or greater

1 - request accepted

2 or greater - request rejected

This argument shall be an integer variable or an integer array element.

3.6 Determing the Current Value of an Eventmark, IREDEM

Execution of a reference to the integer function IREDEM will return the then current value of the specified eventmark. The condition of the eventmark will be unaffected. The form of this function reference is as follows:

IREDEM (e)

where:

e specifies the eventmark, an integer expression

4. EXECUTIVE INTERFACE

4.1 Control of Task Execution

The executive interfaces described in this section provide the facility to control the operation of tasks within the system. Through these external procedures, the execution of tasks may be started, stopped, delayed or synchronized.

4.1.1 Relationship between the Complete and Simplified Forms of the Executive Interfaces

The executive interfaces required for the control of task interrelationships can be complex but in many applications simple interfaces are entirely satisfactory. To provide for both complex and simple applications, the executive interfaces are given in two forms. The first is a complex subroutine reference requiring a large number of arguments and this form provides great flexibility and capability. The second is a set of simple subroutine references with only a few arguments. These simple subroutine references are identical to the complex form with certain arguments taking default values. To make this relationship explicit, the description of each simple form includes the equivalent complex form with the appropiate default argument values.

The executive interface procedures of the standard ANSI/ISA S61.1-1976 section 2 are examples of the simple forms, and for completeness and to emphasis their relationship to the complex form, the definitions are repeated in this standard. There is absolutely no change in the functions of the procedures described in ANSI/ISA S61.1-1976.

4.1.2 Task Identification

Each task identification must specify an executable program, and each initiation creates a new virtual processor. A processor dependent argument p is provided in task initiation calls so that additional information can be specified to the executive program. Such information can include but is not restricted to priority, additional task identification and security information. To distinguish between virtual processors created with the same executable program, the argument p can provide additional information. The executive program may use this information as part of the virtual processor identification. If the argument p is used for identification, then all subroutine references refer only to those virtual processors with the same identification. This identification is specified by the parameter pair i and p. If p is not used for the identification, and if virtual processors exist which were initiated with the same executable program, then subroutine references to this executable program refer to all such virtual processors.

4.1.3 Exception Handling

The argument m, shown below in each of these executive interface procedures, shall be set equal to or greater than two (2) in value for all instances in which the request was not accepted by the executive routine. Individual implementations may specify unique values of m within the allowable range to designate the specific reason for which the request was rejected.

4.2 Starting a Task Immediately or After a Specified Time Delay or Upon an Event Occurrence, SKED

Execution of a reference to the subroutine SKED shall cause the execution of the designated program, after the expiration of the specified time delay or at the desired time of day or upon the occurrence of a specified event. The program thus scheduled becomes known as a task. The actual time resolution obtainable in a specific industrial computer system is subject to the resolution of that system's real time clock. Execution of the designated task will commence at the program's first executable statement. The task is placed in the state PENDING, the task's virtual processor comes into existence and the task is known and identifiable to the real processor. If a task is initiated by an event occurrence, the executive program will decrement the eventmark. The form of this procedure reference is:

CALL SKED (i, p, s, e^1, t^1, n, t^2, c, t^3, e^2, e^3, m)

where:

i specifies the program to be executed.

The argument is either:

a) an integer expression, or

b) an integer array name, or

c) a procedure name

The processor shall define which of the above three forms is acceptable.

p the definition and usage of this argument is processor dependent. An integer expression.

s specifies the selection between event initiated execution and time dependent initiation; an integer expression evaluated as follows:

= 1 start immediately

= 2 start at time of day indicated by t^1

= 3 start when eventmark specified by e^1 becomes ON

= 4 start at time t^1 or when the eventmark specified by e^1 becomes ON, whichever is first

= 5 start after time period t^1 has expired

= 6 start after time period t^1 or when the eventmark specified by e^1 becomes ON, whichever is first

e^1 specifies the eventmark; an integer expression

t^1 designates an integer array name whose first eight (8) elements contain the time at which the specified program is to be executed.

These elements are as follows:

First element - Hours (0 to 23)

Second element - Minutes (0 to 59)

Third element - Seconds (0 to 59)

Fourth element - Milliseconds (0 to 999)

Fifth element - basic units of the system real time clock.

Sixth element - Day (0-31)

Seventh element - Month (0-12)

Eighth element - Year from AD 0

n specifies a time phasing by which the starting time of the designated task would be deferred from the time stated in t^1 or the event specified by e^1. This variable must be an integer expression evaluated as follows:

= 1 no deferred start

= 2 start at the next occurrence of the time of day specified by t^2 after the time or event specified by t^1 or e^1 depending on s, occurs.

= 3 start after a period of time specified by t^2 from the occurrence of time t^1 or event e^1 depending on the value of s.

t^2 specifies the time of day or incremented time for the deferral of the start of a task as specified by n; this must be an integer array name as described in t^1.

c specifies the cyclic execution of the task; an integer expression evaluated as follows:

= 1 no cyclic execution (run once only).

= 2 cyclic execution each time period as specified in t^3.

= 3 cyclic execution each time the eventmark specified by e^2 becomes ON.

t^3 specifies the time period for cyclic execution; this must be an integer array name as described in t^1.

e^2 specifies the eventmark for cyclic execution; an integer expression.

e^3 specifies the eventmark to be turned ON upon a cyclic overrun condition; an integer expression. Note: cyclic execution overrun can only be set by a SKED request with c = 2 or 3.

m is set on return to the calling program to indicate the disposition of the request as follows:

The value must be 1 or greater

1 - request accepted

2 or greater - request rejected

This argument shall be an integer variable or integer array element.

4.2.1 Cause a Program to Cycle at a Stated Frequency, CYCLE (a shortened form of SKED)

Execution of a reference to the subroutine CYCLE shall cause the execution of the designated program immediately and shall cause successive executions to occur after specified periods of time. The cyclic period is defined as the period between successive initiations of the specified program. Resolution of time is subject to the limitations of the system's real time clock. If a cyclic overrun condition occurs, the action is processor dependent and no eventmark is set. The form of this procedure reference is:

CALL CYCLE (i, p, t^3, m)

where:

i specifies the program to be executed.

The argument is either:

a) an integer expression, or

b) an integer array name, or

c) a procedure name

The processor shall define which of the above three forms is acceptable.

p the definition and usage of this argument is processor dependent; an integer expression

t^3 specifies the cyclic time period; this must be an integer array name whose first eight (8) elements contain the time period between successive initiations of the specified program. These elements are as follows:

First element - Hours (0 to 23)

Second element - Minutes (0 to 59)

Third element - Seconds (0 to 59)

Fourth element - Milliseconds (0 to 999)

Fifth element - basic units of the system real time clock.

Sixth element - Day (0-31)

Seventh element - Month (0-12)

Eighth element - Year from AD 0

m is set on return to the calling program to indicate the disposition of the request as follows:

The value must be 1 or greater

1 - request accepted

2 or greater - request rejected

This argument shall be an integer variable or integer array element.

An execution of a reference to CYCLE has the same effect as a reference to SKED with the following values of the remaining arguments:

$s = 1$

$e^1 = 0$

$t^1 =$ (don't care)

$n = 1$

$t^2 =$ (don't care)

$c = 2$

$e^2 = 0$

$e^3 = 0$

4.2.2 Cause a Program to Execute Each Time a Stated Eventmark Becomes ON, CON

Execution of a reference to the subroutine CON shall cause the designated program to become associated with the specified eventmark such that the task shall perform one execution each time the stated eventmark changes from OFF to ON. The executive program will decrement the specified eventmark. The task shall begin from the first executable statement. If a cyclic overrun condition occurs, the action is processor dependent and no eventmark is set. The form of this procedure reference is:

CALL CON (i, p, e^2, m)

where:

i specifies the program to be executed.

The argument is either:

a) an integer expression, or

b) an integer array name, or

c) a procedure name

The processor shall define which of the above three forms is acceptable

p the definition and usage of this argument is processor dependent; an integer expression

e^2 specifies the eventmark; an integer expression

m is set on return to the calling program to indicate the disposition of the request as follows:

The value must be 1 or greater

1 - request accepted

2 or greater - request rejected

This argument shall be an integer variable or integer array element.

An execution of a reference to CON should have the same effect as an execution of a reference to SKED with the following values of the remaining arguments:

$s = 3$

$e^1 = 0$

t^1 = (don't care)

$n = 1$

t^2 = (don't care)

$c = 3$

t^3 = (don't care)

$e^3 = 0$

4.2.3 Starting a Program Immediately or After a Specified Time Delay, START

Execution of a reference to the subroutine START shall, after the expiration of the specified time delay, cause the execution of the designated program. The actual time delay obtainable in a specific industrial computer system is subject to the resolution of that system's real time clock. Execution of the designated program will commence at the program's first executable statement. The form of this call is:

```
CALL START(i,j,k,m)
```

where:

i specifies the program to be executed.

The argument is either:

a) an integer expression or

b) an integer array name or

c) a procedure name

The processor shall define which of the above three forms is acceptable.

j specifies the minimum length of time, in units as specified by k, to delay before executing the program. If the value of j is zero or negative, the request program will be run as soon as permissible. This argument shall be an integer expression.

k specifies the units of time as follows:

0 - Basic counts of the system's real time clock

1 - Milliseconds

2 - Seconds

3 - Minutes

This argument shall be an integer expression.

m is set on return to the calling program to indicate the disposition of the request as follows:

The value must be 1 or greater

1 - Request accepted

2 or greater - Request not accepted

This argument shall be an integer variable or integer array element.

An execution of a reference to START has the same effect as a reference to SKED with the following value of the arguments:

i = same as START

p = 0 (default value)

s = 5

e^1= 0 (default value)

t^1= an array of eight elements which values are dependent upon the value of the k argument of START.

if: k=0, $t^1(5)$=j

k=1, $t^1(4)$=j

k=2, $t^1(2)$=j

k=3, $t^1(2)$=j

all other elements of t^1 are equal to zero

n = 1

t^2= (don't care)

c = 1

t^3= (don't care)

e^2= 0 (default value)

e^3= 0 (default value)

m = same as START

4.2.4 Starting a Program at a Specified Time, TRNON

Execution of a reference to the subroutine TRNON shall cause the designated program to be executed at a specified time of day. Execution of the designated program will commence at the program's first executable statement. The form of this call is

```
CALL TRNON(i,j,m)
```

where:

i specifies the program to be executed.

the argument is either:

a) an integer expression or

b) an integer array name or

c) a procedure name

The processor shall define which of the above three forms is acceptable.

j designates an array whose first three(3) elements contain the absolute time of day at which the specified program is to be executed. These elements are as follows:

First element - Hours(0 to 23)

Second element - Minutes(0 to 59)

Third element - Seconds(0 to 59)

This argument shall be an integer array name.

m is set on return to the calling program to indicate the disposition of the request as follows:

The value must be 1 or greater

1 - Request accepted

2 or greater - Requested rejected

This argument shall be an integer variable or integer array element.

An execution of a reference to TRNON has the same effect as a reference to SKED with the following value of the arguments:

i = same as TRNON

p = 0 (default value)

s = 2

e^1= 0 (default value)

t^1= an array of which the first three values correspond to the three elements of the array j argument of TRNON. The other values of t^1 are as follows:

$t^1(4) = 0$

$t^1(5) = 0$

$t^1(6) - t^1(8)$ = current system date

n = 1

t^2= (don't care)

c = 1

t^3= (don't care)

e^2= 0 (default value)

e^3= 0 (default value)

m = same as TRNON

4.3 Descheduling a Task

Execution of a reference to the subroutines CANCEL, DCYCLE and DCON shall immediately disassociate the execution of the specified program with the initiating conditions previously specified. The task associated with the program will be removed from the state PENDING, the task's virtual processor will become non-existent and the task will become unknown to the real processor.

4.3.1 Cancelling a Task Scheduled for Future Execution, CANCEL

Execution of a reference to the subroutine CANCEL shall immediately remove the specified task from the time schedule of pending executions. There is no effect on current executions. The form of the call is as follows:

```
CALL CANCEL (i,p,m)
```

where:

i specifies the task to be removed from the time schedule. The argument is either:

a) an integer expression, or

b) an integer array name, or

c) a procedure name

The processor shall define which of the above three forms is acceptable.

p The definition and usage of this argument is processor dependent; an integer expression.

m is set on return to the calling program to indicate the disposition of the request as follows:

The value must be 1 or greater

1 - request accepted

2 or greater - delay as specified has not occurred

This argument shall be an integer variable or integer array element.

4.3.2 Removing a Task from Cyclic Scheduling, DCYCLE

Execution of a reference to the subroutine DCYCLE shall immediately remove the specified task from the list of cyclic time based executions but allows the execution currently in the RUNNING State, if any, to be completed. The form of the call is as follows:

```
CALL DCYCLE (i,p,m)
```

where:

i specifies the task to be removed from cyclic scheduling. The argument is either:

a) an integer expression, or

b) an integer array, or

c) a procedure name

The processor shall specify which of the above three forms is acceptable

p the definition and usage of p is processor dependent; an integer expression.

m is set on return to the calling program to indicate the disposition of the request as follows:

The value must be 1 or greater

1 - request accepted

2 or greater - request rejected

This argument shall be an integer variable or integer array element.

4.3.3 Removal of a Task From Time or Event Based Schedule, DCON

Execution of a reference to the subroutine DCON shall immediately remove the specified task from all future scheduled executions and/or association with any event occurrence. The form of the call is as follows:

```
CALL DCON (i,p,m)
```

where:

i specifies the task name to be descheduled.

The argument is either:

a) an integer expression, or

b) an integer array name, or

c) a procedure name

The processor shall define which of the above three forms is acceptable.

p the definition and usage of p is processor dependent; an integer expression

m is set on return to the calling program to indicate the disposition of the request as follows:

The value must be 1 or greater

1 - request accepted

2 or greater - request not accepted

This argument shall be an integer variable or integer array element.

4.4 Delay Execution of the Program, DELAY

Execution of a reference to the subroutine DELAY will cause suspension of the program execution until the specified event occurs or the specified time of day or time period has elapsed. Whichever occurrence happens first will cause the program to resume execution at the statement following the call. If the program was DELAYed for an eventmark to change state from OFF to ON the executive program will automatically decrement the specified eventmark prior to the resumption of execution. The task will be moved from the state RUNNING to the State VOLUNTARY SUSPEND. A task may only delay itself and cannot delay another task or be delayed by another task. The format of the call is:

```
CALL DELAY (s, t, e, m)
```

where:

s specifies the selection between event initiated end-of-delay and time dependent initiation; an integer expression evaluated as follows:

1 - delay ends at the time of day specified by t

2 - delay ends at the end of the time increment specified by t

3 - delay ends when the eventmark specified by e changes from OFF to ON

4 - delay ends whichever occurs first: time increment specified by t, or the eventmark specified by e changes state.

5 - delay ends whichever occurs first: the time of day specified by t, or the eventmark specified by e changes state.

t designates an integer array name whose first eight (8) elements contain the time at which the specified program is to be executed. These elements are as follows:

First element - Hours (0 to 23)

Second element - Minutes (0 to 59)

Third element - Seconds (0 to 59)

Fourth element - Milliseconds (0 to 999)

Fifth element - Basic units of the system real time clock

Sixth element - Day (0-31)

Seventh element - Month (0-12)

Eighth element - Year from AD 0

e specifies the eventmark; an integer expression

m is set on return to the calling program to indicate the disposition of the request as follows:

The value must be 1 or greater

1 - request accepted

2 or greater - requested rejected

This argument shall be an integer variable or integer array element.

4.4.1 Suspend execution of a program for event occurrence, WAITE

Execution of a reference to the subroutine WAITE will cause suspension of the program execution until the specified event occurs. The program will resume execution at the statement following the call. The executive program will automatically decrement that specified eventmark prior to the resumption of execution. The format of the call is:

CALL WAITE (e,m)

where:

e specifies the eventmark; an integer expression

m is set on return to the calling program to indicate the disposition of the request as follows:

The value must be 1 or greater.

1 - request accepted

2 or greater, request rejected

This argument shall be an integer variable or integer array element name.

Execution of a reference to WAITE shall have the same effect as an execution of a reference to DELAY with the value of s=3 and no particular value of t.

4.4.2 Delaying Continuation of a Program, WAIT

Execution of a reference to the subroutine WAIT shall return after a delay of a specified length of time. The form of this call is:

```
CALL WAIT(j,k,m)
```

where:

j specifies the length of time in units as specified by k to delay before returning to the calling procedure. If the value is zero or negative, no delay will occur. Limitations of the implementation shall not cause the precise time to be less than requested. This argument shall be an integer expression.

k specifies units of time as follows:

0 - Basic counts of the system's clock

1 - Milliseconds

2 - Seconds

3 - Minutes

This argument shall be an integer expression.

m is set on return to the calling program to indicate the disposition of the request as follows:

The value must be 1 or greater

1 - Request accepted

2 or greater - Delay as specified has not occurred.

This argument shall be an integer variable or integer array element.

Execution of a reference to WAIT shall have the same effect as an execution of a reference to DELAY with the value of s=2, the array t having a value as specified below depending upon the argument k of WAIT, and no particular values of e.

if: k = 0, t(5) = j

k = 1, t(4) = j

k = 2, t(3) = j

k = 3, t(2) = j

4.5 Abort a Program, ABORT

Execution of a reference to the ABORT subroutine will cause the named program to immediately terminate execution and to disassociate itself from all forms of scheduled activity whether based upon event occurrence or time. The task is removed from the states RUNNING or VOLUNTARY SUSPEND. The task's virtual processor becomes non-existent and the task becomes unknown to the real processor. The form of the call is:

```
CALL ABORT (i,p,m)
```

where:

i specifies the program to be aborted.

The argument is either:

a) an integer expression, or

b) an integer array name, or

c) a procedure name

The processor shall define which of the above three forms is acceptable

p the definition and usage of p is processor dependent; an integer expression.

m is set on return to the calling program to indicate the disposition of the request as follows:

The value must be 1 or greater

1 - request accepted

2 or greater - request rejected

This argument shall be an integer variable or integer array element.

4.5.1 FORTRAN Program Termination, STOP or END

Normal termination of a program in FORTRAN is accomplished by execution of the STOP statement or alternatively the END statement. Tasks may abort themselves by execution of the STOP or END statements or may execute a reference to the ABORT subroutine where the argument references are of the task itself.

APPENDIX A

This appendix is not part of the ISA Standard S61.3 but is included to facilitate its use.

Considerations leading to Industrial Computer System FORTRAN Procedures for the Management of Independent Interrelated Tasks.

A.1 Historical Development

This standard is a direct outgrowth of the International Purdue Workshop on Industrial Computer Systems whose goals are:

To make the definition, justification, hardware and software design, procurement, programming, installation, commissioning, operation, and maintenance of industrial computer system more efficient and economical through the development of standards and/or guidelines on an international basis.

The Workshop formed several committees to achieve its objectives. The FORTRAN Language Committee was charged with the task of preparing a set of Industrial Process standards compatible with standard FORTRAN.

This standard is the result of that committee's work.

A.2 Criteria Used in Developing FORTRAN Standards

The committee assessed the status of FORTRAN as used in the industrial environment and followed the guidelines below in the devlopment of the standards:

(1) The standards whould cover features commonly used by existing industrial computer systems.

(2) The standards should be easily implemented by most vendors.

(3) The standards should follow the syntax and intent of FORTRAN as defined by ISO R1539-1972.

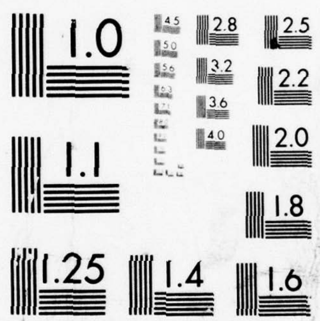
(4) The standards should not restrict the future evolution of FORTRAN language.

The development of FORTRAN language standards is presently the responsibility of ANSI/X3J3. In order that ISA standards comply with the ANSI standards as far as possible, external-procedure references were used rather than direct changes or additions to the syntax of FORTRAN. This does not imply that this is the only way to provide these features, nor does it exclude the possibility or desirability that ANSI will develop the language syntax to perform these and other related functions.

AD-A068 267 PURDUE UNIV LAFAYETTE IND PURDUE LAB FOR APPLIED IND--ETC F/G 9/2
MINUTES 1978 SPRING REGIONAL MEETING INTERNATIONAL PURDUE WORKS--ETC(U)
JUN 78 N00014-78-C-0127

UNCLASSIFIED NL

3 OF 4
AD
A068267



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

APPENDIX B

This Appendix is not part of the ISA Standard S61.3 but is faciliatate its use.

NOTES BY SECTION

B.1 Section 1 Notes

The exclustion from this standard in Table I of al sharing files, sharing variables or arrays, or sendi between tasks does not mean that these features of tas are unimportant, but that it is considered pr standardize such features at the present time; par these features depend on the design of the executi However a standard for file access and the con contention exists (ISA S61.2-1978) and these procedu used to ensure the sharing of data and the passing of an orderly manner.

There is no provision in the standard for BOOLEAN o eventmarks. Such features may be provided by th program or be available at system generation time processor dependent. The lack of such features is not discourage their inclusion by system suppliers considered premature to standardize on such features a

Eventmarks are not variables within a users program be read, set, cleared or tested with the interfaces d Section 3. The desirability of a system setting an ev the logical OR or the logical AND of two events is obv is not possible for a users program to perform operations as in PL/I but no such feature is available FORTRAN.

A cyclic execution overrun can occur for vario Improper program design can result in a cyclic overru in the task's virtual processor, i.e., the task can its work before its next invocation even if all requir are available. However, even if a cyclic overrun wi in a task's virtual processor, it can occur in the rea because the executive program and real processor ma sufficient resources to the task to avoid a cyclic ov time-oriented cyclic executions, good system design understanding of the performance of the executive prog processor can ensure that cyclic overruns will almost but for event-oriented cyclic executions where th between events can be very small, even good system d eliminate the likehood of cyclic overruns.

B.2 Section 2 Notes

In this section the states of a task are defined i task's virtual processor. The concept of a virtual pr

introduced into this standard as a method of description of task interrelationships. The relationship between tasks in a real processor involves not only inter-task considerations but also consideration of the performance capability and design of the executive program. For example at any instant a task may not be executing in the processor for several reasons such as:

- Priority of another task
- Lack of resources--memory, disk, etc.
- Waiting for completion of input/output
- Waiting for the occurrence of an event

The first three of the above reasons depends on the particular processor and associated executive program, only the last involves inter-task considerations. It is not the intent of this standard to define the performance, capability, attributes or design of executive programs. Therefore the concept of a virtual processor was introduced to provide a mechanism which accurately separates those features of task performance which are dependent on the executive program from those dependent only on the relationships of individual independent interrelated tasks.

B.3 Section 3 Notes

The eventmark defined in this section provides capability for handling events. The use of events is the responsibility of the system designer but two (2) common usages can be envisaged.

An event can be considered to be an interrupt, either hardware or software, which exist in most industrial computer systems. The occurrence of an interrupt can cause the executive routine to set an eventmark and the procedure references defined in Sections 3 and 4 of this standard provide all the necessary features to control computer usage based on interrupts.

An event can also considered to be a Dijkstra P-V semaphore and can be used for the control of critical resources and the resolution of contention for such resources. The well-known one-writer-many-readers problem can be solved by using the eventmarks as semaphores.

In the use of eventmarks it is the system designers responsibility to ensure that no lock-out conditions occur. There is no provision in the standard to ensure that inadvisable use of the eventmarks will not cause undesirable consequences.

B.4 Section 4 Notes

B.4.1 Specification of Program "Name"

This standard is a permissive standard in that it does not prescribe how the executive will respond to external procedure

references nor does it describe how the information is passed to the executive routine. In particular, the argument "i" in CALL SKED (4.2), CALL DSKED (4.3) and CALL ABORT (4.4) has three forms, an integer expression, an integer array name or a program name, only one of which is permissible in any particular program. This restriction is a necessary consequence of the requirements of the FORTRAN language that any argument which is an external procedure reference be of a defined type;

integer expressions, integer array names and program names are different types (Section 8.4.2 of ISO-R1539-1977). It is also a consequence of the FORTRAN language that if the argument is a program name, this name must appear in an EXTERNAL statement (Section 7.2.15 of ISO-R1539-1972).

Examples of the use of the argument i as an integer expression are:

```
--CALL ABORT (7,M)

--DATA J/7/
  CALL ABORT (J,M)

--DATA J/2HAB/
  CALL ABORT (J,M)
```

An example of the use of the argument i as an integer array name is:

```
INTEGER XYZ
DIMENSION XYZ(3)
DATA XYZ(1),XYZ(2),XYZ(3)/2HAB,2HCD,2HEF/
CALL ABORT (XYZ,M)
```

An example of the use of the argument i as a procedure name is:

```
EXTERNAL ABCD
CALL ABORT (ABCD,M)
```

Interchangeability of programs between different processors is reduced by permitting three possible types for this argument but the committee feels it is premature to standardize to achieve full interchangeability in this area.

B.4.2 Technical Discussion on Section 4

The subroutine SKED provides great flexibility in the initiation of a task but requires a large number of arguments. The simpler forms provide more limited capability but with few arguments. The features of SKED provide the ability to design complicated task initiation programs which are commonly required at system start-up time so that all the actions can be done in a single start-up program. The same functional result can usually be achieved with a complicated interrelated set of tasks using the simpler forms but

the simplicity of a single program is lost and the problems of documentation and system maintenance increased.

For a non-cyclic initiation, a task exists in the PENDING state only once. For a cyclic task, a task returns to the PENDING state many times; indeed following the initiation of its execution on transfer to the state RUNNING, a cyclic task will automatically reappear in the PENDING state. Thus for cyclic tasks there exists a set of occurrences that can be called "future PENDING". This distinction is of no importance to the act of initiation but can be of importance when considering the problems of descheduling.

Descheduling has different considerations that those of scheduling. In the initiation of a task an association is created between the program identification (name, number, etc.) and the task. One program can be associated with many different tasks and this causes no problems for SKED, but descheduling requires someway to identify which task is to be terminated. Another consideration is whether to deschedule the PENDING task or the future PENDING or both. A third consideration involves the method of change from PENDING to RUNNING such as event or time. Thus the decision on descheduling involves three areas:

- The identification of the task

- Choice of PENDING and/or future PENDING

- Method of change from PENDING to RUNNING

The standard provides for choices in the method of change but always deschedules all PENDING and future PENDING tasks. It is possible to envisage methods by which an executive program could distinguish between these two situations but such distinctions are not a common feature of existing executive programs and the committee felt it was premature to standardize in this area.

The problem of task identification can be solved in three ways:

a) Deschedule all tasks associated with the program.

b) At scheduling time, the user assigns a task identification in use for descheduling.

c) At scheduling time, the executive program returns to the user a task identification for use in descheduling.

The procedure in a. is simple to implement and is the one used by many existing executive programs. If the user provides a task indentification as in b. it is necessary for the executive program to keep this information, but it can be assumed that the user's program will know the identification of the task which is to be descheduled. If the executive program provides the task identification as in c. it is necessary to provide some mechanism for the program that desires to deschedule the task to know the identification assigned by the executive program. This requires a

mechanism for shared variables or intertask communication which is not included in the scope of this standard.

The standard provides for the descheduling of all tasks associated with a given program if the executive program will not accept a user task identification. If the executive program will accept a user defined task identification (defined in the argument "p") then this identification can be used for descheduling. However, the construction of the argument "p", the rules for its use, and the exact actions of the executive program are not defined. No provision is made in this standard for task identification by the executive program.

APPENDIX A-IX

AD HOC MICROCOMPUTER COMMITTEE - PURDUE AMERICAS

1. Keiles, Yoel, Fault Tolerance in Distributed Microprocessor Control System Architecture by Redundancy, ISA Conference, Niagara Falls, New York, October 1977.

FAULT TOLERANCE IN DISTRIBUTED MICROPROCESSOR CONTROL SYSTEM ARCHITECTURE VIA REDUNDANCY

Yoel Keiles

Honeywell Inc.
Ft. Washington, PA

Abstract

Super-availability in real time control. Architectural trade-offs in implementation of super-availability. Fault detection techniques. Data base transfer and switching of input and output signals and a description of fault tolerant implementation in TDC 2000.

Introduction

The primary reason for fault tolerance in a microprocessor distributed structure is to achieve super-availability. Such availability cannot be obtained with an equivalent analog architecture. The super-availability concept will ensure continuation of optimal control strategy. Fault tolerance will smooth plant control. It will limit the operator usage of the customary manual mode to start-up or override decisions, and most important it will prevent failures that may have catastrophic consequences to property and human life. The major differences between process control and distributed data processing is that in process control, the I/O signals are an intermix of both analog and digital signals. And more important, the reliability of a process control system is much more significant due to the actual plant operation being affected by a failure.

Architectural Consideration

An inherent characteristic of a distributed microprocessor based control system is characterized by its fault limiting ability. In other words, faults do not propagate throughout the system and are localized to the UniMicroprocessor subsystem. Adding fault tolerance means that when a fault is detected, a substitute subsystem will be automatically inserted into the system and the faulty subsystem disconnected without disturbing the process being controlled. At a convenient later time, the faulty subsystem will be repaired. While the subsystem is being repaired, the system continues to control the processes. When the proper operation of the repaired subsystem is restored, the operator will initiate an automatic return to the original system structure before the fault occurred, again without disturbing the process being controlled. On the other hand, instead of performing the last step, it is possible to "rotate" the repaired subsystem as a spare. However, the added hardware and software complexity prevents this approach from being worthwhile. A hundred fold reliability improvement is possible with the exact number depending on the specific implementation. The above described super-availability concept achieves almost unity availability.

Redundancy in a fault tolerant system has to achieve these functional tasks:

- o Fault detection
- o Data base transfer
- o Switching of inputs and outputs
- o Insertion of spare substitute
- o On-line repair and manual initiation of an automatic switch back from spare substitution state to the original state before the fault occurred.

At this point, a few definitions are in order. A first level is the Input/Output level at which the inputs (analog/digital) and outputs are connected to the field. This level can operate independently since it is microprocessor based, however, it also communicates with a second level. The second level is defined as the man/machine interface. The second level includes both the information necessary to control the process and the communication link (Data Hiway - See Figure 1) to the first level.

One possible architecture to achieve the above described tasks is to have a device in the second level (e.g., a computer or an Operator Station) that periodically scans the distributed lower level devices. An example of such a first level device is a digital controller. When a fault is detected, another identical device in the first level is substituted for the faulty subsystem. One disadvantage of such an architecture is that to achieve recovery of the data base from the faulty subsystem, the local power, the communication link, and the microprocessor base must remain operational at all times. This type of operation encompasses a great deal of hardware. ... is that the data base

PRECEDING PAGE BLANK - NOT FILMED

transfer via the communication link causes an unpredictable load that might interfere with other real time operations of the system. Because of these disadvantages, it seems that a separation between first level and second level redundancy is desired. Reinforcing the need for such a separation is the fact that the first level is the front line of control and necessitates a fast response time while the second level has a more moderate requirement. This paper addresses the super-availability concept in the first level.

The separation between the first and second level results in the following advantages:

- o Minimum operational hardware.
- o External powering of the critical hardware.
- o Higher diagnostic resolution prevents a transfer if spare subsystem is also malfunctioning.
- o Sharing switching mechanism hardware overhead between many devices to improve economy.
- o Shorter local communication link cable results in greater noise immunity and a higher transfer data rate due to higher bandwidth.
- o All first level devices operate independently if the communication link to second level malfunctions.
- o Local changing of addresses for establishing a communication link between first level and second level devices addresses makes man/machine interface simple, i.e., the operator window to the process is transparent with or without substitution.

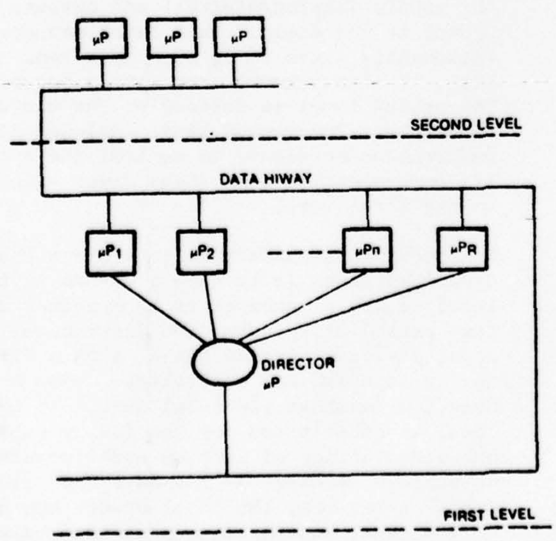


FIGURE 1 - FAULT TOLERANCE
DISTRIBUTED CONTROL SYSTEM ARCHITECTURE

Fault Detection Techniques

Concurrent checking hardware and self-diagnostics in the UniMicroprocessor subsystem is a way to achieve fault detection, as follows:

- Measurement of the state of the UniMicroprocessor system.
  A check sum is calculated by the microprocessor and stored performing an exclusive or of all register, and memory location whose content is not dependent on data. The result of the exclusive is compared with the stored answers. For each program module, two such check sum calculations are recommended (software).

- Entry/Exit Flags in each Software Module.
  Upon entry, each software module establishes a unique number. During the execution of the module, the present state of the number is compared with the desired state. In this way, reasonable program flow is checked (software).

- Write Protection on Programs.
  In regular operation only, read operations will be performed from this memory section. Performing a write indicates an illegal operation and stops the processor (hardware).

- Memory Parity Protection of the Random Access Memory.
  Any single bit failure will be detected. This detection mechanism is independent of the microprocessors. Upon failure, the diagnostics will determine if this is a hard failure or a soft failure by re-running this section of the program (hardware).

- Reasonableness Tests of Data.
  One can only check that the known results of operation are being followed, but reasonableness tests are extremely helpful (software).

- Diagnostic Test of the UniMicroprocessor.
  The diagnostic test of the microprocessor can be divided into two sections - an execution of the instruction set and execution of data manipulation through all paths of the machine (software).

- Equipment Failure.
  The watchdog timer not serviced <u>or</u> (logical) there is a power supply failure <u>or</u> (logical) an input device is not serviced <u>or</u> an output device is not serviced (hardware and software), a result from an equipment failure.

- Traps to Detect Unreasonable Program States.

- Test, That a Given Task is in Reasonable Boundary of Time (software).

Data Base

The data base in continuous process control, usually includes set-point values, tuning parameters, operating mode and configuration. A redundant local data base has the advantage of

increasing reliability and permitting on-line repair. The main memory is word organized to facilitate fast parallel data transfer within the microprocessor. The redundant memory is bit organized to prevent any need for syncronyzation and to facilitate fast serial transmission. The serial storage architecture lends itself to up-coming technology of charge couple devices or bubble memory. The redundant memory has dual accessability when operating in the serial mode and is logically isolated from the microprocessor. Keeping the redundant data base locally with each microprocessor has the advantage that if there is a power blackout, the redundant data base can be used to restart the process when the power is restored.

The transfer of data, the scanning to determine which one of the many uniprocessors is malfunctioning, the continuous self-diagnostics on the local data base, the communication control and initiation of the switching of the I/O matrix are achieved by another microprocessor in a star tee configuration (Figure 1) referred to as the director. The director status output signal is transmitted to the second level to indicate the state of the system and to identify where on-line repair is necessary.

Switching the Inputs and Outputs

Because the input/output signals on the first level may be analog, no loss of accuracy can be tolerated in the switching matrix. An input is usually a voltage source (1-5 volt), and the output is a current source (4-20 milliamperes). As previously mentioned, the director microprocessor initiates the switching. Consequences of failure in the input/out switching matrix is the most critical element in the redundancy mechanism.

Description of Fault Tolerant Implementation in TDC 2000

An array of one to eight microprocessor controllers is locally supervised by a director microprocessor to achieve an uninterrupted automatic control system. The director detects the failure to the operator in the second level and switches in a reserve controller identical to the primary controllers.

Each controller receives analog signals and every 1/3 of a second converts them to digital signals as inputs to selected algorithms, then converts the computational results to analog signals for an output. Up to eight process loops can be controlled by each controller. A read-only memory (ROM) contains an executive and a 28 algorithm application library from which the computational recipes are selected in configuring a slot. The random access memory (RAM) is the working memory area which is backed up by a second memory to provide data for the reserve controller. The redundant second memory uses CMOS memory (4Kx1). This redundant memory can be powered from a battery or from the director in a triple diode matrix. The redundant memory of the reserve controller is identical to that in the primary. Each of the controllers has direct memory through the Data Hiway, i.e., a coax cable that ties together first and second levels. The director pseudo-executive and other programs are in firmware. The Input/Output matrix is switched via reed relays.

The local data base transfer communication is via standard TTL transmitter/receiver circuits and is configured in a radial architecture. The director has a direct memory (DMA) circuit that permits communication to the second level via the Data Hiway. The director also has a local man/machine interface as well as a remote interface via the hiway.

The director has 4 modes of operation:

- o normal
- o backup
- o test
- o reset

In the normal mode, the director monitors all primary controllers and eight remote manual command lines which can cause the director to switch to the backup mode. It also exercises confidence routines. In the backup mode, the microprocessor goes through all of the sequence to activate the reserve controller. A test mode is initiated either remotely or locally, to perform all hardware/software diagnostics. If during this mode a primary controller fails, the director will automatically go to backup mode. A reset mode is used when the confidence routines detect a director problem or in the initial power-on sequence.

All the above modes are displayed and initiated locally (LED) or remotely (CRT).

Conclusions

Local star tee configuration architecture in the first level (I/O) makes super-availability very economical and provides less intervention of the process control operator under abnormal conditions.

References

Robert J. Bibbero, "Microprocessors in Instruments and Control", ---Page 252-260, Wiley, 1977.

SECTION V

TUTORIALS

1. Pingle, Thomas C., SPIDER, A Hierarchical Industrial Manufacturing and Control System, Spring Regional Meeting International Purdue Workshops on Industrial Computer Systems, Purdue University, West Lafayette, Indiana, April 1978.

2. Malagardis, Nicolas, Report on Standards - Development and Processing of a Standard, Spring Regional Meeting International Purdue Workshops on Industrial Computer Systems, Purdue University, West Lafayette, Indiana, April 1978.

3. Bachman, Charles W., Distributed System "Reference Model", Spring Regional Meeting International Purdue Workshops on Industrial Computer Systems, Purdue University, West Lafayette, Indiana, April 1978.

4. Graube, Maris, Process Interface Comparisons, Spring Meeting International Purdue Workshops on Industrial Computer Systems, Purdue University, West Lafayette, Indiana, April 1978.

5. Messare, Remsi R., Long Range Functional Requirements and Design Criteria for Operator Consoles, Spring Regional Meeting International Purdue Workshops on Industrial Computer Systems, Purdue University, West Lafayette, Indiana, April 1978.

TUTORIAL NUMBER 1

SPIDER, A HIERARCHICAL INDUSTRIAL MANUFACTURING AND CONTROL SYSTEM

Mr. Thomas C. Pingle
Western Electric Company
Lisle, Illinois

Presented at the Purdue Americas Meeting

INTRODUCTION

SPIDER is the acronym for a Computer Aided Manufacturing (CAM) project responsible for controlling and supporting manufacturing processes at multiple Western Electric manufacturing locations. It is a hierarchical computer network which interfaces the Computer Aided Design (CAD) systems of Bell Laboratories and Western Electric Standards Engineering to manufacturing facilities on the factory floor for high technology electronic switching system products (Figure 1). SPIDER, which stands for Shared Production Interactive Data Bases for Error Reduction, is best remembered by the visual image it suggests, that being a central host (body) controlling a web of manufacturing needs by connecting to numerous satillites via direct communication links (legs).

SPIDER has been developed at the Northern Illinois Works of the Western Electric Company in Lisle, Illinois. The Lisle plant is a prototype, early production facility for new products being designed at the Indian Hill facility of Bell Laboratories in nearby Naperville. As new products pass through the development phase into a mature, stable design, responsibility for the continued manufacture of the products is transferred to other plants for high volume/low cost production. The SPIDER network has developed to not only manage the technical data support for this new product introduction, but also to facilitate the transfer to, and continued manufacturing support in, a full scale production environment.

The impetus for a computer network such as SPIDER came from two trends within the Bell System which I am sure are common to other industries. The first of these was the proliferation of stand alone, minicomputer controlled manufacturing and test processes. The second, not unrelated, trend was the increased use of Computer Aided Design systems within Bell Laboratories. These conditions were compounded by the prototype, early production status of the Northern Illinois Works and its accompanhing high rate of design change activity, and thus was dictated the need for networking capabilities to administer and control the flow of "machine readable" data from the CAD outputs to the machine readable input requirements of the minicomputer. An increasing strain was beingplaced on the capabilities of our manufacturing engineers just to keep up with the data. Thus began the evolution of the SPIDER network which currently supports manufacturing operations at two other factories - Dallas, Texas and Oklahoma City, Oklahoma - in addition to our Lisle plant. The network also includes links to the sending systems at Bell Laboratories and Western Electric Standards via our divisional Warrenville Data Center, and a backup link to the corporate time sharing data center in Kingsbridge, New Jersey (Figure 2).

A host satellite hierarchy concept was selected for SPIDER as one best suited to our intended applications. The host machine provides the processing capabilities, file handling, and memory of a large computer. The minicomputer satellites provide the real time, interrupt driven capabilities required in a process control environment. Where possible, we have extended the hierarchy to a third level where the mini acts as a "mini host" to multiple in-house designed U-CONs or microprocessors, each of which controls an individual machine or process. To a large extent the minicomputers which previously were used in stand alone dedicated manufacturing facilities have been "hooked up" to the network. This addition of communication and data links to the host processor thus provides the satellites with the facilities of a multimode system designed to be used concurrently by many users who wish to perform many and varied tasks. The host in the hierarchy is currently configured to support 180 simultaneous jobs. The mini-hosts can support up to 16 each. Access is provided to the network via hard wired data lines or dial-up telephone (Figure 3).

RESPONSIBILITIES

Computer Aided Manufacturing connotes a wide and varied range of activities to most people. The SPIDER network has been developed with very specific functions and responsibilities in mind. There are two major functions which direct the activities of the development team. The first is the responsibility for administering technical manufacturing data used to drive computerized manufacturing facilities and to provide diagnostic assistance to the people operating those facilities. This requires establishing procedures to transfer the data, as needed, to the people in the process or the satellite processors.

The network is currently supporting a full range of facilities in the manufacture of telephone switching equipment. These include assembly and test facilities from the lowest level of apparatus assembly, subassembly test and wiring verification through a complete system test. While all of these processes receive data in one form or another from the host machine, not all are directly connected to the network. As the network evolved, we had hoped to connect all of the stand alone processes into the network. Some of these processes, however, were commercial turn key systems or developed within Western Electric as stand alone support systems. These types of facilities generally are not designed with communications capabilities. If there were several machines of the same type where this case existed or if the facility required massive quantities of data which changed frequently, it was economically feasible to develop communications interfaces.

For example, this was done for the automatic wiring machines and the wiring verification facilities. Where this was not feasible, data support was provided in the traditional manner, i.e. magnetic tape, paper tape, etc.

As facilities were added to the network, the area of diagnostic assistance using the technical manufacturing data became a significant consideration. This area was not initially recognized in our network planning. The application of CAM techniques to the factory was successful and was changing the priorities of the factory problems. Traditional queues were shifting and a means of providing data for diagnostic assistance became a major data management need. The requirement was essentially one of fast information retrieval for the people in the process. The technical manufacturing data is now presented on CRTs on request with keys and other aids to manufacturing operators.

The second primary responsibility of the network is to record, analyze and report on troubles found throughout the manufacturing and installation processes. This provides a means to evaluate design problems, manufacturing problems, and to monitor quality levels. The ability to capture this information provides a feedback mechanism to the various functional organizations that leads to correction or improvement in current designs and manufacturing methods.

This function has been designed with two levels of detail. The first level concerns defect recording, reporting and analysis across the entire manufacturing and installation processes. At selected locations in the process one hundred per cent of the defects reported are recorded. This subsystem is called TRAP (Trouble Reporting and Analysis Procedures) and has three primary purposes:

1. To provide a historical data base for future study and analysis which is available to interested internal organizations and also is provided to Bell Telephone Laboratories.

2. To provide operating shop personnel with the ability to interactively retrieve historical failure modes. In some testing processes problem solutions are difficult to diagnose and require a considerable amount of time. We have assisted in this area by providing information on past failure modes and their solutions. It provides a starting point for troubleshooting and as new problems and solutions are found and recorded, becomes almost a probabilistic model.

3. To trigger the next level called Product Characterization. The computer provides a continual analysis of available TRAP

data in specific areas and identifies exceptional or trending conditions which require special attention. Product Characterization involves detailed analysis to determine the cause(s) of the exception. When the cause has been identified and corrected, the detailed analysis is stopped. This analysis is costly and with the ability to turn it on and off only as needed minimizes this cost.

PHILOSOPHY

The SPIDER host has become the central repository of manufacturing data generated by the Bell Telephone Laboratories CAD systems and the Western Electric Standards Engineering Organization. The associated data bases are centrally controlled and administered yet available to all facilities connected to the network. The SPIDER network could be defined as a centralized network based upon the host-satellite concept or perhaps a hierarchical network. However, the classical definitions of network types (if there are any agreed upon classical definitions) do not fit the SPIDER network. The host machine does not control the satellites in our configuration. The satellite machines are conceptually independent processors which interrupt the host machine when service is required. This is contrary to one of the essential characteristics of multilevel hierarchical systems; specifically, the priority of action of right of intervention of the higher level system. This implies an importance of action and goals of the higher system over the lower levels. However, our reason for existence is to support factory operations. They are our primary customer, and therefore, have top priority. Our philosophy emphasizes priority for the very lowest level in the network.

Since the satellite processors are dedicated to specific facilities, we have tried to force as much processing as possible out to the satellites. This serves two purposes: to save resources of the host machine, and to fully utilize the computing power of these dedicated processors. Each satellite is configured with a disk for storage of control software and enough manufacturing data to guarantee the manufacturing process can continue for a period of time in case of a host machine failure.

Pushing the processing to the satellite processors and even beyond allow us to control our input and more effectively manage the distribution of the technical data. We can specify a single data format per job class and postpone any data transformations required by facility differences to the lowest level in the hierarchy. By maintaining a single format in our total host data base, we gain a flexibility and ease of introducing new facilities into the network by concerning ourselves only with the low level interface.

CONCLUSIONS

The SPIDER system is a hierarchical industrial manufacturing and control system providing the link between Computer Aided Design and minicomputer manufacturing applications. It provides centralized data administration and communications with manufacturing processes not only locally at Northern Illinois Works, but also with our Oklahoma City and Dallas locations. In addition, it provides manufacturing engineers with a tool to improve existing or develop new Computer Aided Manufacturing techniques. Software development tools such as simulators, cross assemblers and debugging aids are available. We have developed what we feel is an open-ended network philosophy that provides the ability to change, update, or expand as our manufacturing techniques evolve.

We did not spend years designing a large total integrated system and then attempt to implement it. We are implementing our system in an existing manufacturing environment. This environment is dynamic and continually changing. We have attempted to evolve a hierarchy of subsystems to create an overall CAM network. A major concern is to always be aware of the needs of our users. We feel we have been successful. There have been major benefits in the reduction of redundant developments, better machine control and utilization, faster and easier assistance to the people in the process and control of delivering information to the factory floor and fast feedback of troubles found throughout our processes.

We are continuing to evolve and do not see the end. Access to information on a timely basis is the obvious need of management and production employees. We are constantly looking at the needs of our users and these needs tend to change as our network evolves. Some of these needs may become more important than our current thrust on facilities and feedback. If this is so, our philosophy enables the network to respond to these new priorities.

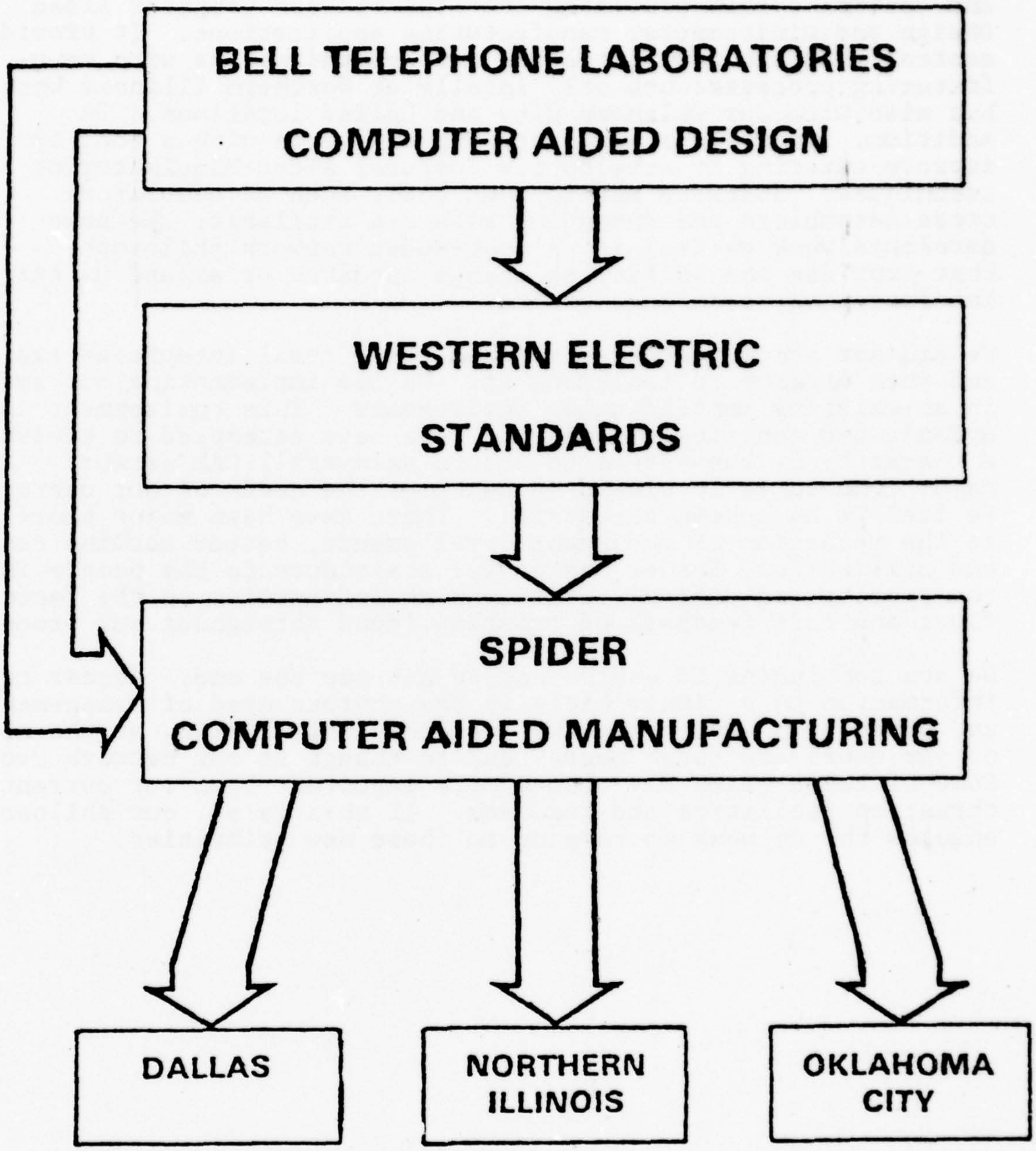

BELL TELEPHONE LABORATORIES
COMPUTER AIDED DESIGN
WESTERN ELECTRIC
STANDARDS
SPIDER
COMPUTER AIDED MANUFACTURING
DALLAS
NORTHERN ILLINOIS
OKLAHOMA CITY

FIGURE 1

SPIDER Network

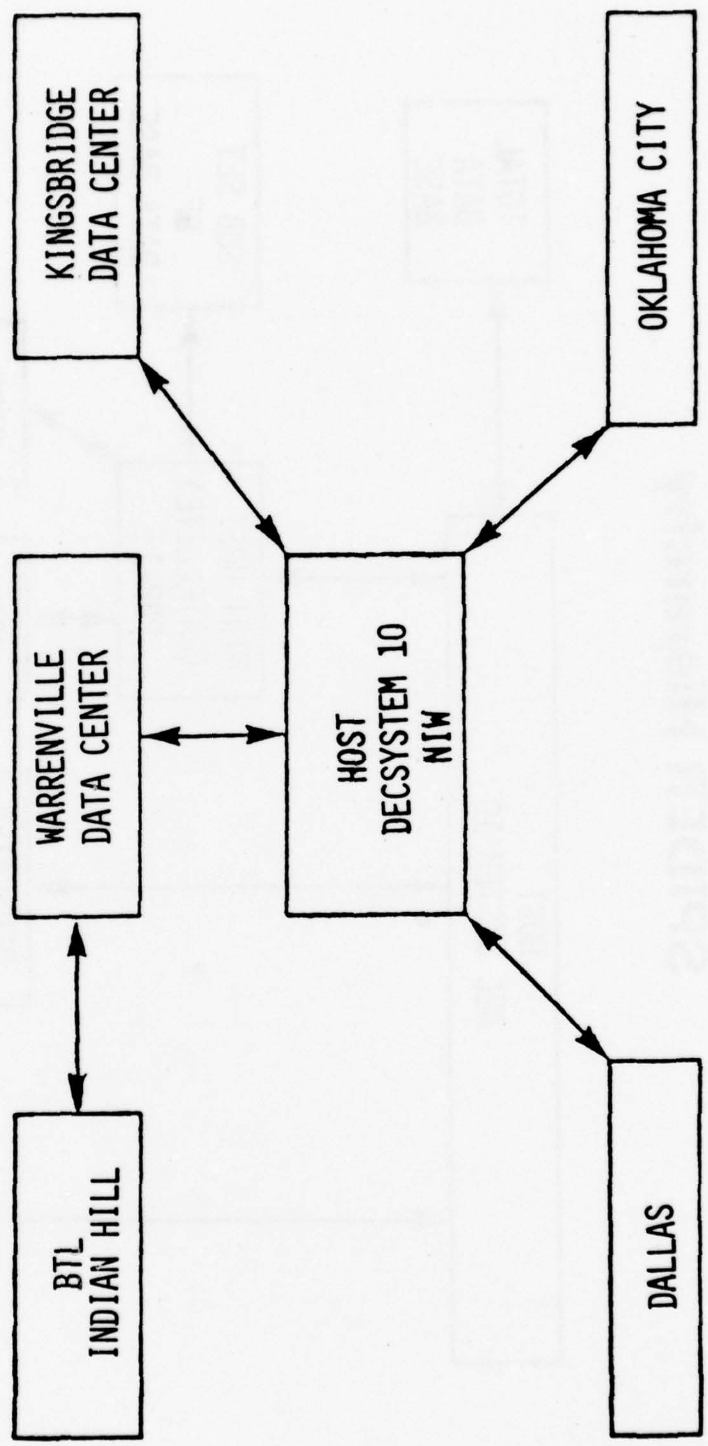


FIGURE 2

SPIDER Hierarchy

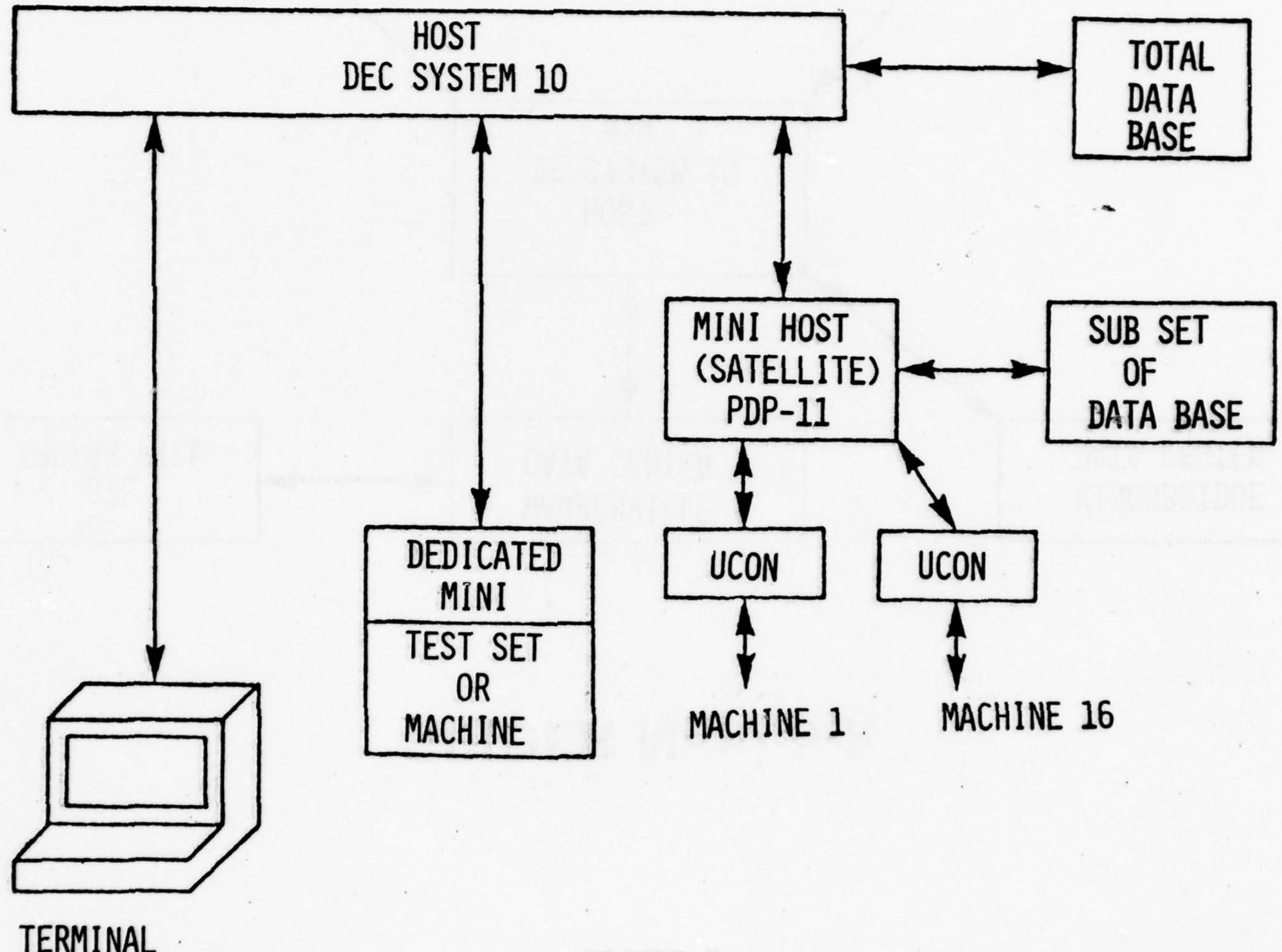


FIGURE 3

TUTORIAL NUMBER II

REPORT ON STANDARDS - DEVELOPMENT AND PROCESSING OF A STANDARD

Mr. Nicolas Malagardis
Director BNI
BP 105
Domaine de Voluceau
F78150 Rocquencourt
LeChesNay
FRANCE

Presented at the Purdue Europe Meeting

* STANDARDIZATION

by Nicolas E. Malagardis

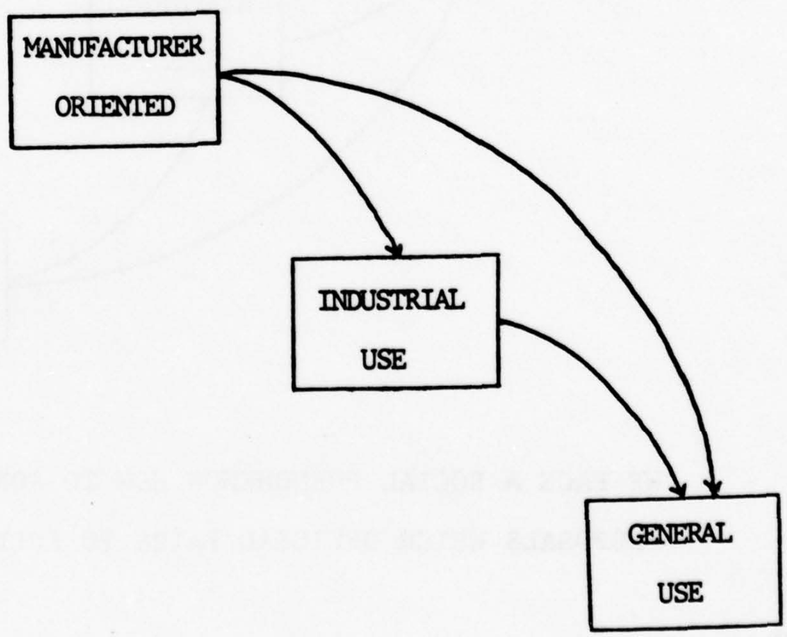
REPORT ON STANDARDS

DEVELOPMENT AND PROCESSING OF A STANDARD

.IDENTIFICATION OF NEEDS

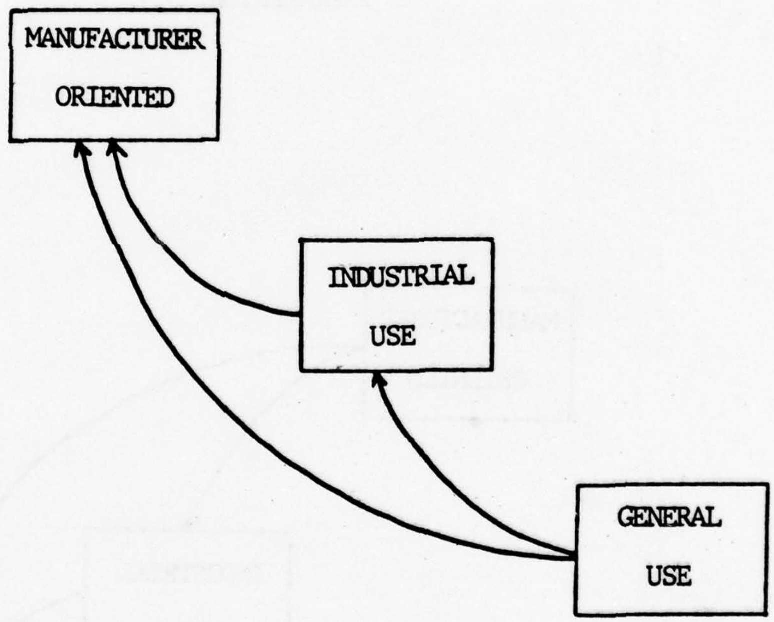
COMPUTER INDUSTRY

INDUSTRIAL USE



.GROWING USER COMMUNITY

TENDENCY TO FORMULATE STANDARDS FROM OUTSIDE TO THE COMPUTER INDUSTRY



WE FACE A SOCIAL PHENOMENON HOW TO FORMULATE BEST PROPOSALS WHICH OFFICIAL PATHS TO FOLLOW

.DEVELOPMENT IS BASED UPON THE PREMISE OF COOPERATION AND TECHNICAL CONTENT REPRESENTS A CONSENSUS OF THE PARTIES INVOLVED.

PHILOSOPHY OF CONSENSUS IMPOSES RESPONSIBILITY UPON THE ORGANISATIONS WHICH FORM THE FRAME WORK WITHIN WHICH STANDARDS CAN BE INITIATED, DEVELOPED AND APPROVED.

ORGANISATIONS MUST DEVELOP PROCESSES METHODS AND OPERATING PROCEDURES THAT GUARANTEE THAT THIS CONSENSUS HAS BEEN REACHED.

.PROCESSING CONSISTS OF THREE PHASES

PLANNING

Proposed scope

Justification of the proposal

Functions expected to be performed

Expressed interest and support

DEVELOPMENT

Formation of a responsible body

Technical work formal and informal technical papers

Approval within the responsible body

APPROVAL

Receipt of the formulated standard at a secretariat of a national or international mandated organisation

Processing establishes now wide acceptance of the proposed standard.

.CRITERIA FOR SELECTING STANDARDISATION PROJECTS

Organisation or responsible committee must ensure that

The proposed standard is needed

It accepts preparation and maintainance

There exists a validation body accepting preparation and administration of a validation

.ESTABLISHING THE NEED FOR A STANDARD

Primary justification economical use of expensive resources

Hints that computer installations will be using the standardised facility beyond a threshold of 10%

Adoption of the standard would world wide save 1% of total cost resulting from proceeding

Existing standard facilities do not cope in this area.

.BENEFITS EXPECTED

- Better software/hardware systems if some of their interfaces become standardised
- Easier recruitment and training of staff
- Flexibility in interchangeability of hardware
- Portability in a greater percentage of applications
- Reduction of software development costs.

. PROBLEMS

- Coordination of projects

  Increasing number of projects requiring cross representation

  Increasing number of committees establishing parallax programs of activities.

- How to proceed?

  Retrospective VS prospective standardisation that is consolidation of existing techniques satisfying past and present practices and needs.

  VS

  Identification of future needs preparation of standard techniques avoiding obsolesence of standards by the time they appear.

- Severe criticisms about consensus philosophy in the USA. General accounting office claims loss of millions of $ for not having appropriate. Standards partners not playing a fair game.

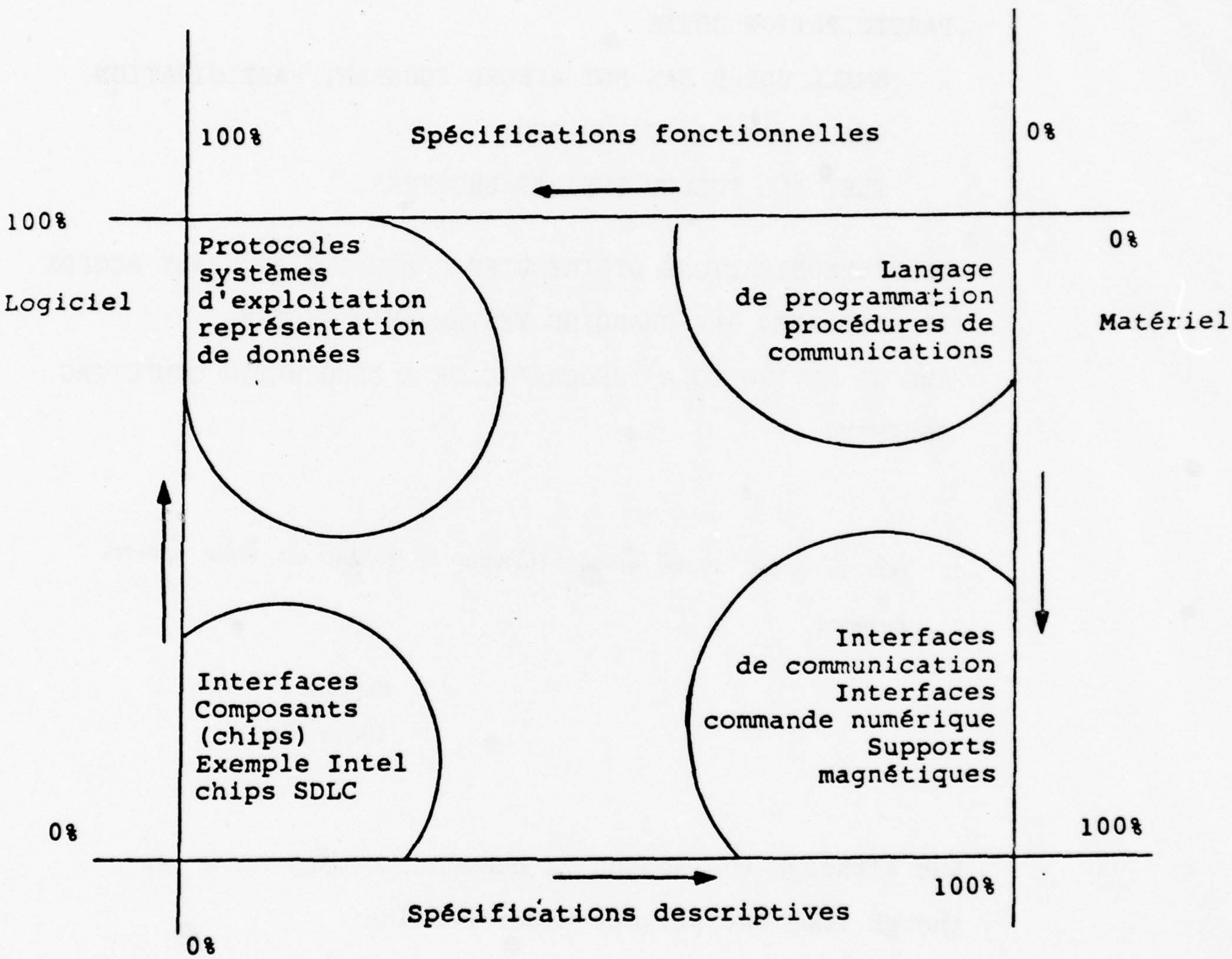
.CLASSIFICATION ATTEMPT

- Descriptive standards
- Functional standards
- Hardware oriented
- Software oriented

Data.

.REPROBLEMS

- Methodology of classification needed
- Standards makers need education
- Computing is still not an exact science.


100%
Spécifications fonctionnelles
0%
100%
0%
Protocoles
systèmes
d'exploitation
représentation
de données
Langage
de programmation
procédures de
communications
Logiciel
Matériel
Interfaces
Composants
(chips)
Exemple Intel
chips SDLC
Interfaces
de communication
Interfaces
commande numérique
Supports
magnétiques
0%
100%
100%
Spécifications descriptives
0%

.PARTICIPATION COSTS

SMALL USERS CAN NOT AFFORD CONSTANT PARTICIPATION.

SMALL MANUFACTURERS TOO.

THEY ALL FOLLOW THE BIG BROTHERS.

.TELECOMMUNICATIONS DISTRIBUTED COMPUTING AND EASY ACCESS TO RESOURCES ARE CHANGING YESTERDAY'S PICTURE.

ARE WE MOVING TO A THEOCRATIC OR A DEMOCRATIC COMPUTING SOCIETY?

τοῦ δέ λόγου κοινοῦ ἔοντος ζώουσιν οἱ πολλοί ὡς ἰδίαν ἔχοντες φρόνησιν.

Heraclite
Diehls Frg 2

But although the Reason is common the many live as though they had private understanding.

.BIBLIOGRAPHY

The world of EDP standards NBS US department of commerce NBSIR 77-1195

Report on standards in computing
National Computing Center
UK MANCHESTER

ISO TC 97 / SC5 doc 386, 387, 388

Various reports on research and technology
U.S. Department of defense

Various articles from the press
mainly Computer Weekly, Computerworld, Datamation.

TUTORIAL NUMBER III

DISTRIBUTED SYSTEM "REFERENCE MODEL"

Charles W. Bachman
Honeywell Information Systems, Inc.
Phoenix, Arizona

Presented at the Purdue Americas Meeting

DISTRIBUTED SYSTEMS "REFERENCE MODEL"

(DRAFT 4)

ANSI/X3/SPARC/Study Group - Distributed Systems

February 20, 1978

78-02-20

ANSI/X3/SPARC/Study Group - Distributed Systems

J. R. Aschenbrenner	IBM
C. W. Bachman	Honeywell (Chairman)
R. M. Brown	CBEMA
M. Canepa	Honeywell
J. Cashin	U. S. Government - USAF
P. Cristy	DEC
R. desJardins	U. S. Government - NASA
J. J. Farley	IBM
J. S. Foley	Burroughs
E. A. Guilbert	TDCC
R. R. Haywood	U. S. Government - NARDAC
S. S. Kimbleton	U. S. Government - NBS
J. P. McGovern	Sperry-UNIVAC
P. L. Petronelli	Boeing Computer Services
V. N. Vaughan	AT&T
J. L. Wheeler	XEROX
G. W. White	U. S. Governmemt - NCS
B. L. Wisner	CDC

78-02-20

OTHER GROUPS KNOWN TO BE INTERESTED IN DISTRIBUTED SYSTEMS ARCHITECTURE

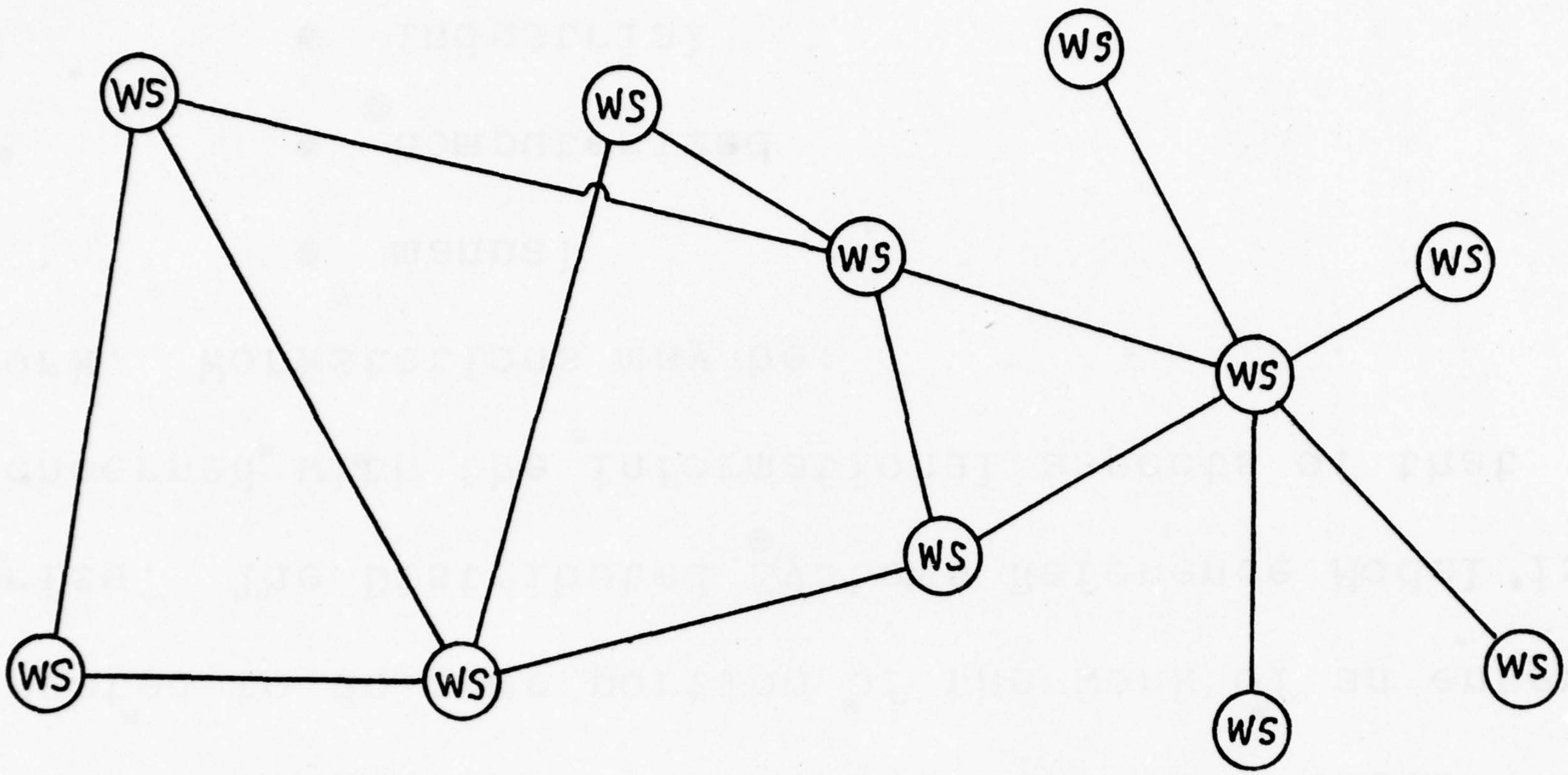
- U.S.A.
  - ANSI/X3S3 Data Communication,
  - ANSI/X3L5 Labels & File Structure,
  - EIA TR30 Data Transmission
- International
  - ISO/TC97/SC6 Working Group 1,
  - ISO/TC97/SC6 Working Group 2,
  - ISO/TC97/SC16 Open Systems Interconnection,
  - CCITT SG VII Public Data Networks,
  - ECMA TC9 Data Communication,
  - ECMA TC23 Systems Architecture,
  - ECMA TC15

78-02-20

INDEX

78-02-20

DISTRIBUTED SYSTEMS CONCEPTS



The distributed systems reference model is based upon modeling information systems as networks of cooperating workstations.
78-02-20

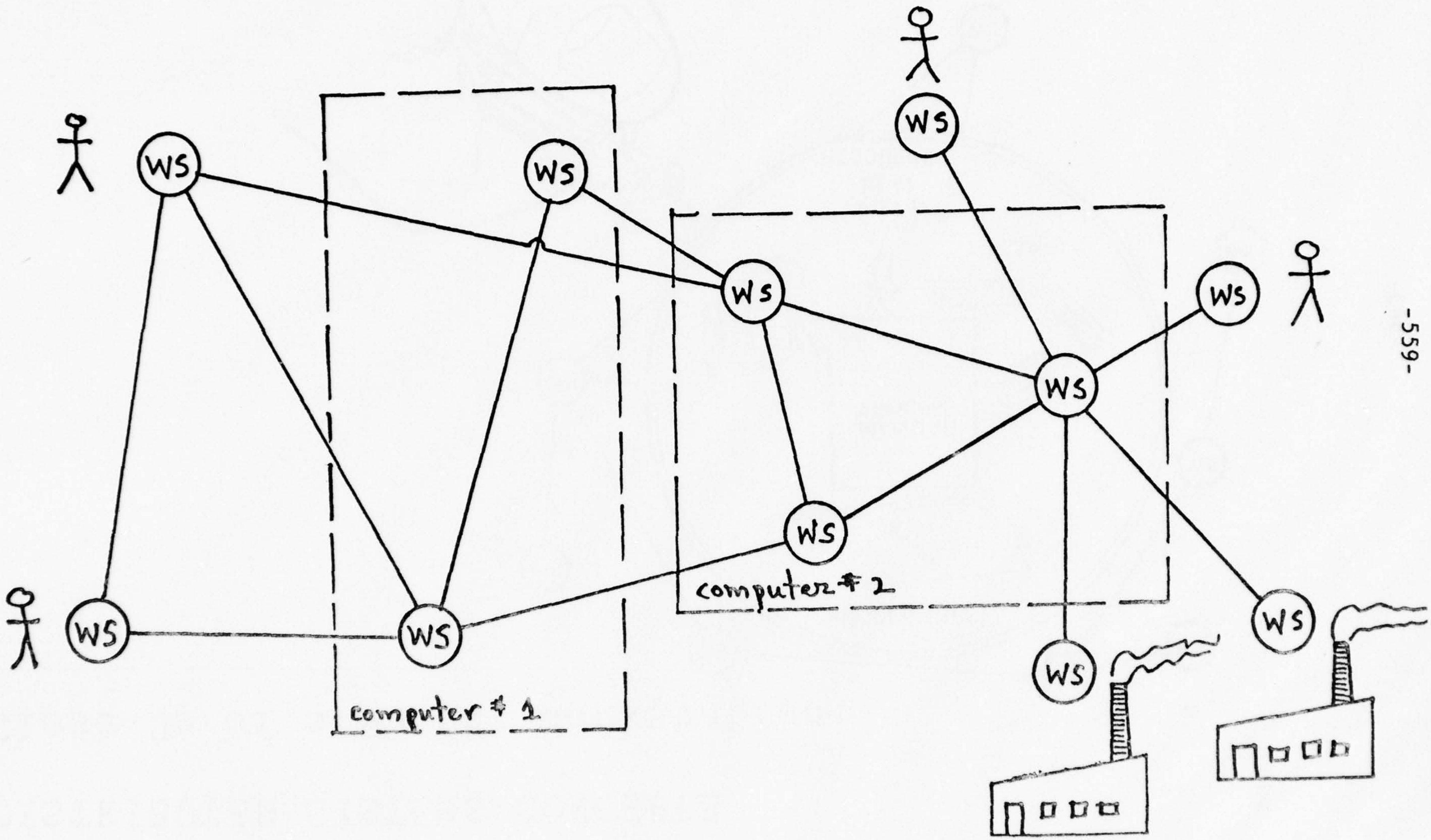
DISTRIBUTED SYSTEMS CONCEPTS

A "workstation" is a cluster of activities created to do some portion of the work of an enterprise. The Distributed Systems Reference Model is concerned with the informational aspects of that work. Workstations may be:

- manual
- computerized
- industrial

DISTRIBUTED SYSTEMS CONCEPTS

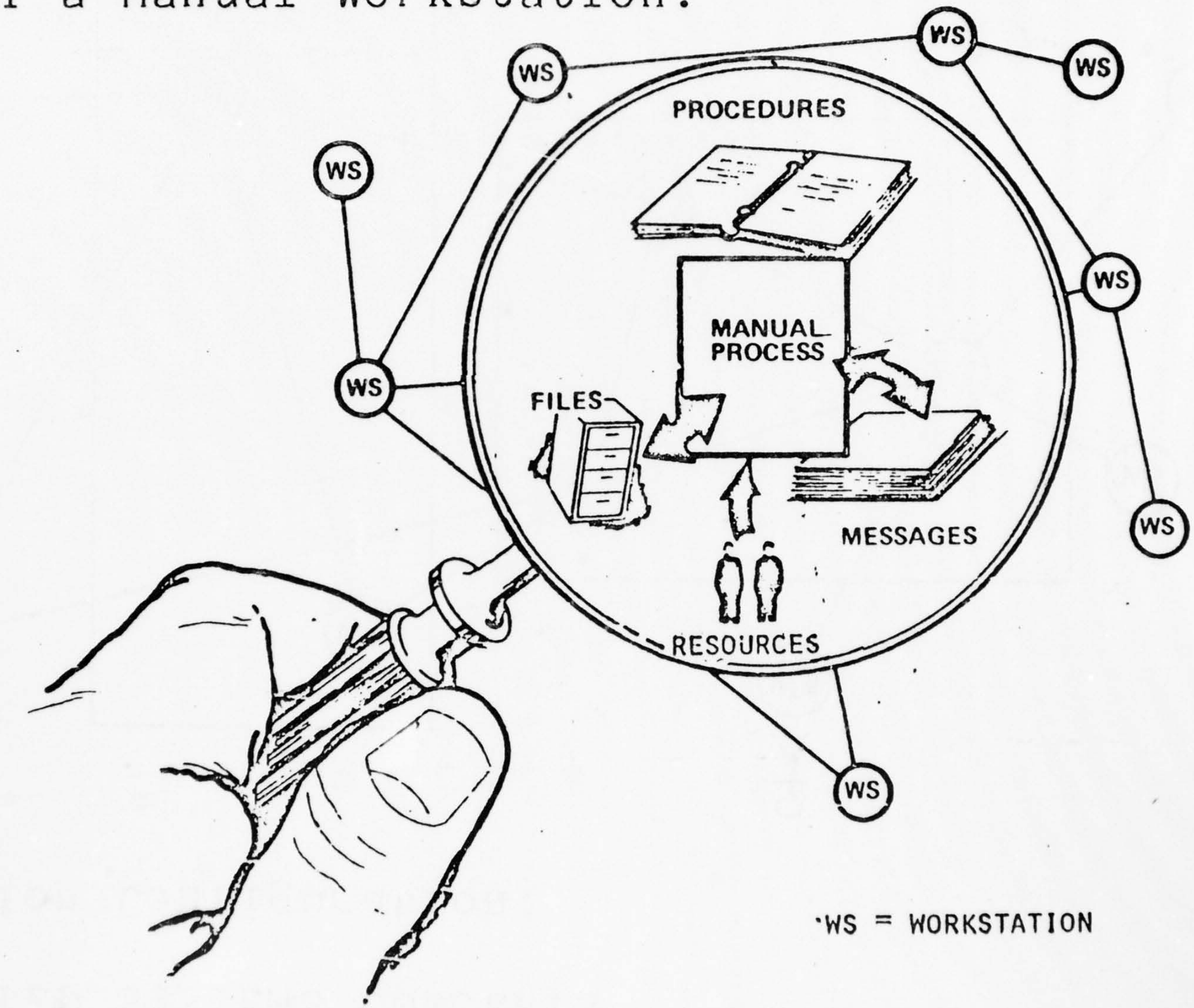
Workstation Configuration:



78-02-20

DISTRIBUTED SYSTEMS CONCEPTS

Close Up of a Manual Workstation:

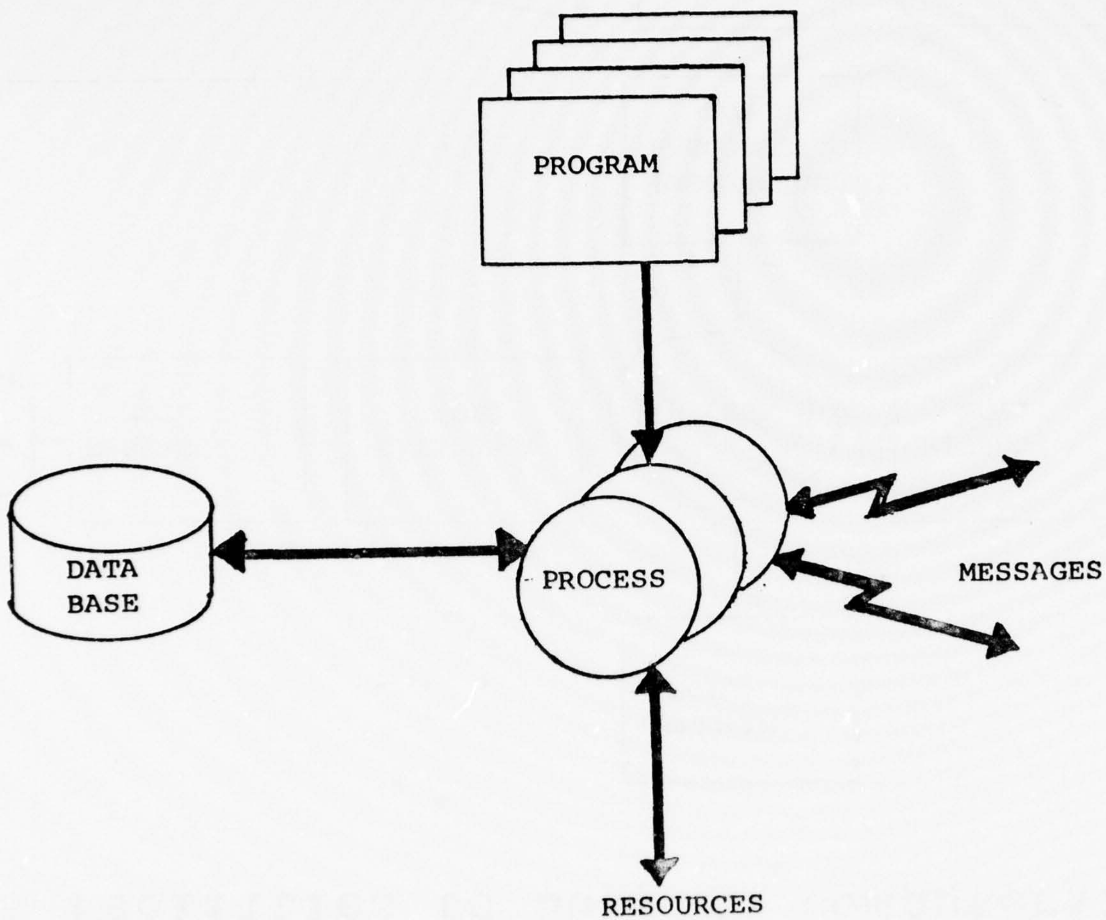


·WS = WORKSTATION

78-02-20

DISTRIBUTED SYSTEMS CONCEPTS

Close Up of a Computerized Workstation:

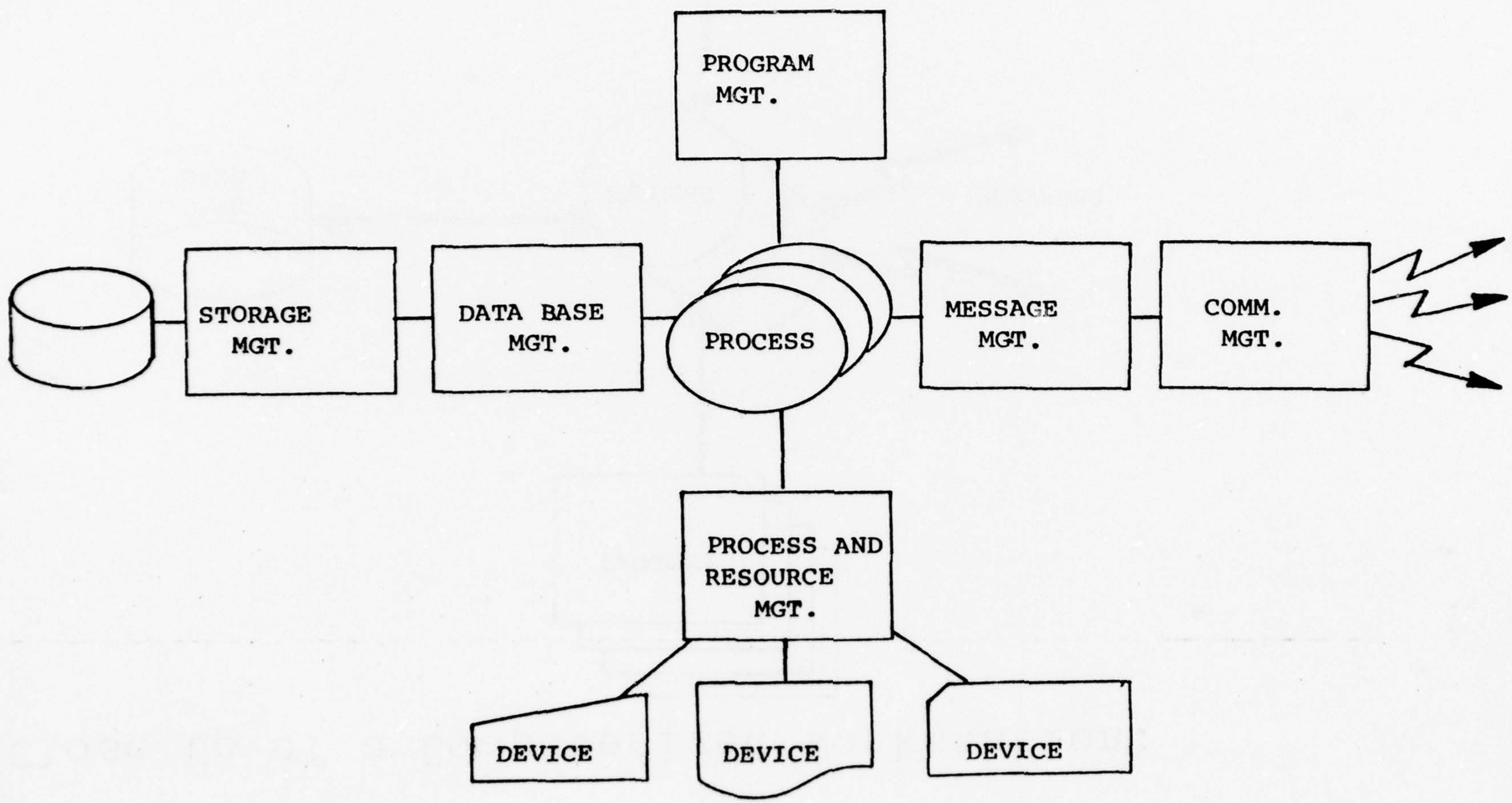


- PROCESSOR
- MEMORIES
- CONTROLLERS
- I/O DEVICES
- etc.

78-02-20

DISTRIBUTED SYSTEMS CONCEPTS

System Facilities to Support Computerized Workstations:



78-02-20

DISTRIBUTED SYSTEMS CONCEPTS

Reference Model:

A reference model is an organized set of:

- functional levels,
- their interfaces and
- their protocols

via which manual, computerized and industrial workstations, when interconnected, can exchange information and thus cooperate in the achievement of the objectives of an enterprise.

78-02-20

DISTRIBUTED SYSTEMS CONCEPTS

Motivation For Establishing Architectural Levels:

The reference model has been created as a set of levels where each level achieves its required functionality partially by itself and partially by making use of the services of the next lower level. The model has been defined such that the services it offers at its interface with the next higher level are independent from the protocols or implementations of that level. Thus a level's particular protocols and implementations are transparent to the user of that level. They can be replaced when appropriate with an alternate protocol and implementation.

DISTRIBUTED SYSTEMS CONCEPTS

Motivation for Specific Levels (continued):

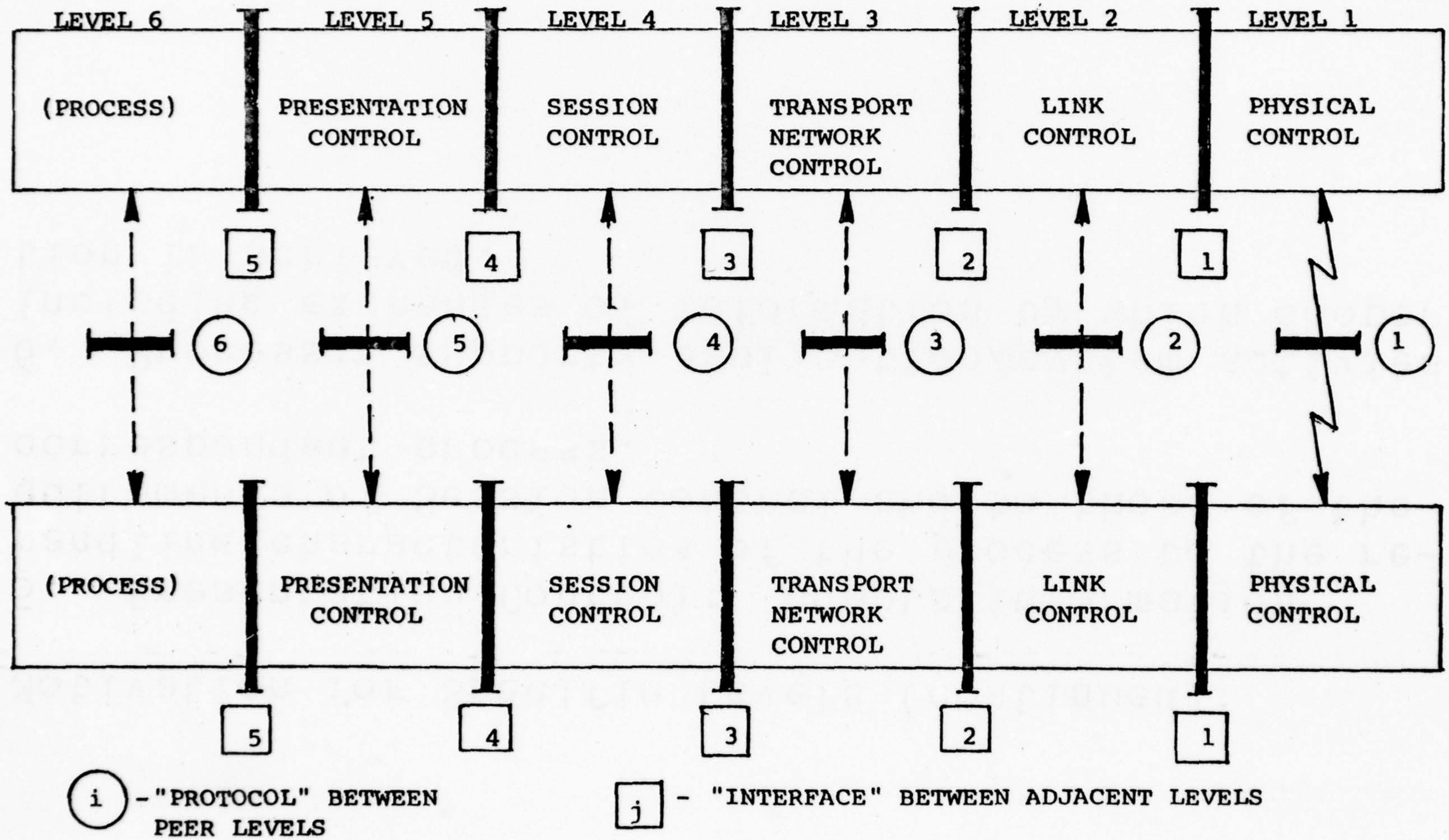
5. Presentation Control: adapts information handling characteristics of the process to the requirements of Session Control and to those of the correspondent process.

6. Process: supports application/system activies including exchanges of information by which cooperation is achieved.

78-02-20

DISTRIBUTED SYSTEMS CONCEPTS

Reference Model:



78-02-20

DISTRIBUTED SYSTEMS CONCEPTS

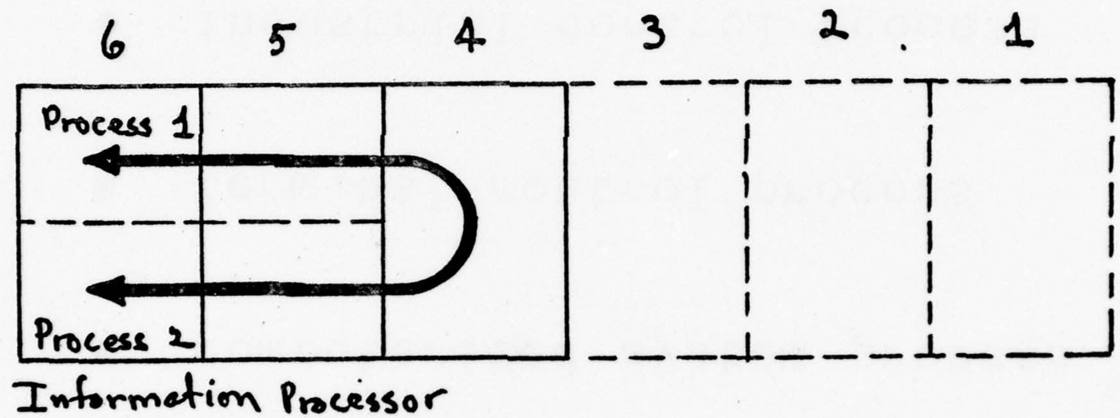
Examples of Process Types:

- computerized application process
- computerized system process
- terminal control process
- industrial control process
- manual process

78-02-20

DISTRIBUTED SYSTEMS CONCEPTS

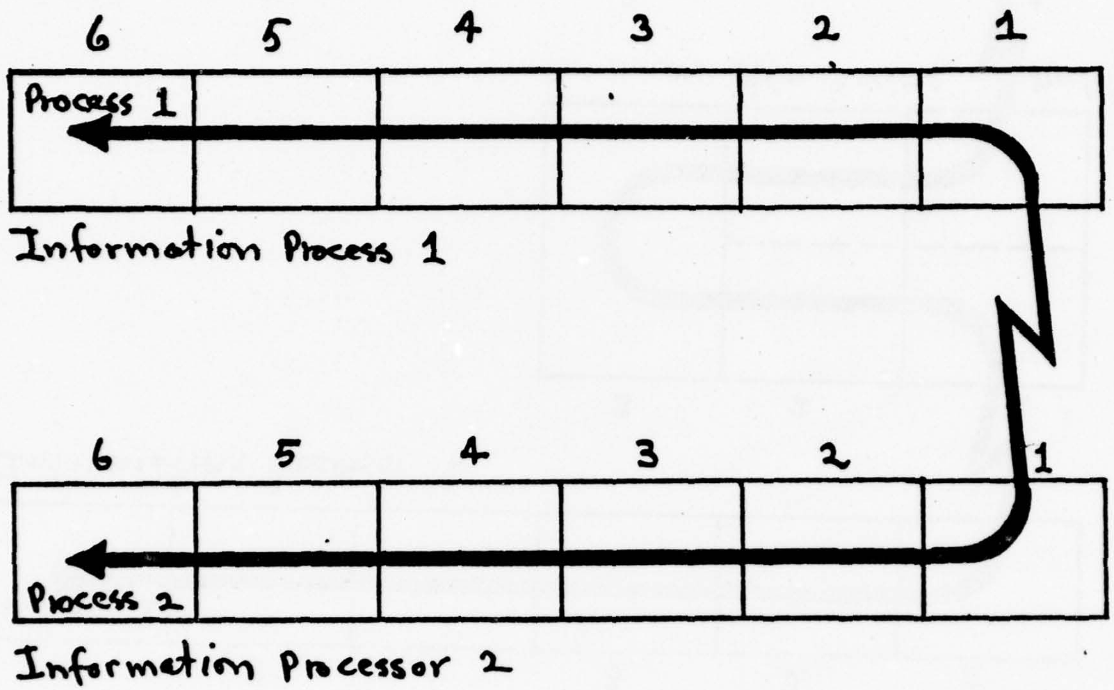
Inter-process Information Exchange Within the Same Machine:



78-02-20

DISTRIBUTED SYSTEMS CONCEPTS

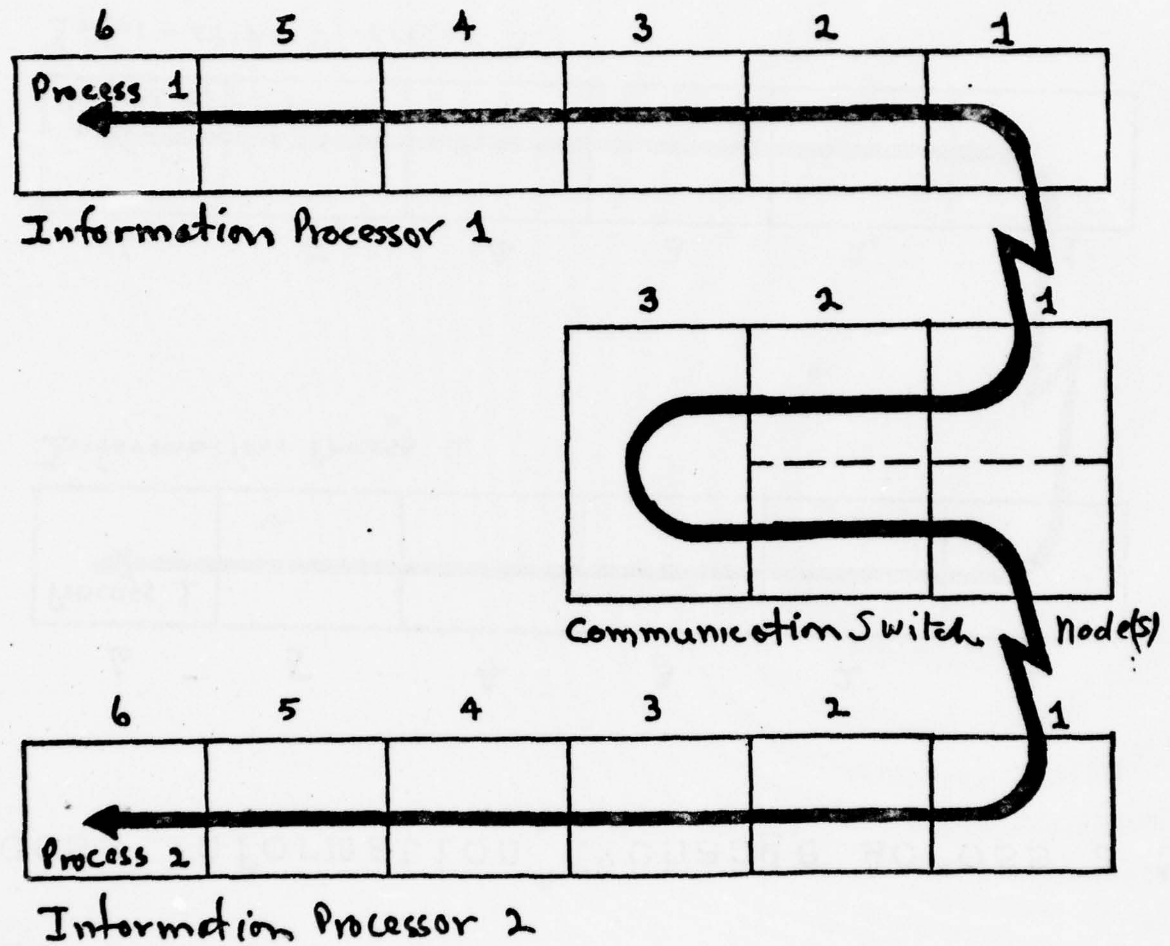
Inter-process Information Exchange Across a Single Link:



78-02-20

DISTRIBUTED SYSTEMS CONCEPTS

Inter-process Information Exchange Across a Communication Network:



78-02-20

DISTRIBUTED SYSTEMS CONCEPTS

Ultimate Goal:

To make every process in the world addressable to one another such that they can exchange information when such exchange appears useful, independent of their location or physical characteristics.

78-02-20

DISTRIBUTED SYSTEMS CONCEPTS

Concepts Applicable at All Levels:

C1 - location: A place where things happen.

C2 - termination: An addressable endpoint at a location.

C3 - connection: A logical association capable of transferring data between terminations.

78-02-20

DISTRIBUTED SYSTEMS CONCEPTS

Concepts Applicable at All Levels (continued)

C4 - data-assurance-unit: A quantity of data whose successful transfer over a connection is acknowledged.

C5 - data-flow-unit: A quantity of data whose transfer over a connection has been authorized.

C6 - data-unit: A quantity of data and control information transferred as a unit over a connection.

C7 - data-mapping-unit: An entity used to map a data-unit of the next higher level onto a data-unit of the current level.

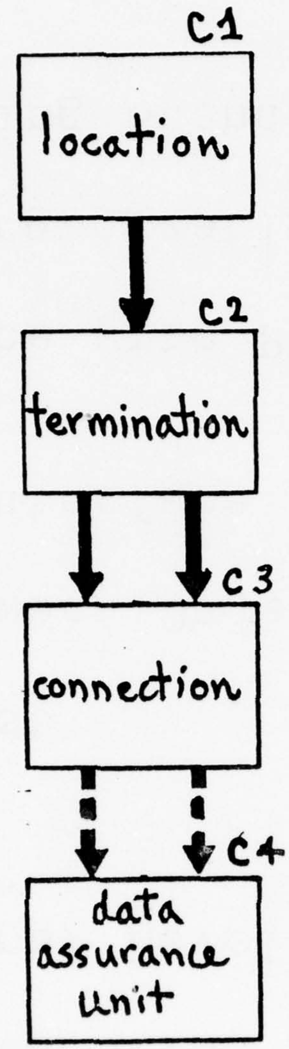
DISTRIBUTED SYSTEMS CONCEPTS

Data Structure Diagrams

A data structure diagram is a graphic tool used to illustrate the structural relationships between the occurrences of different types of entities. A box represents a type of entity. An arrow represents a type of relationship which exists between occurrences of different types of entities. Each occurrence of the entity type at the tail of an arrow is related to zero, one or many occurrences of the entity type at the head of the arrow. Each occurrence of the entity type at the head of a solid line arrow is related to exactly one occurrence of the entity type at the tail of the arrow. Each occurrence of the entity type at the head of a dashed line arrow is related to either zero or one occurrence of the entity type at the tail of the arrow, depending upon the circumstances.

DISTRIBUTED SYSTEMS CONCEPTS

Relationships Between Objects



78-02-20 (data structure diagram)

DISTRIBUTED SYSTEMS CONCEPTS

Outline of Architectural Level Descriptions:

1. Functionality
2. Objects of Interest
3. Relationships Between Objects
4. Services of the Interface with the Next Higher Level
5. Protocol (Interface with Peer Level)
6. Standardization Organizations
7. Examples of Existing Standards

78-02-20

DISTRIBUTED SYSTEMS CONCEPTS

Intent with Regard to Standardization:

It is the current feeling that all the protocols level 1 through level 6, are subject to standardization as soon as a viable specification is prepared which supports the functionality of that level. At some levels, alternate protocols may be appropriate.

Only those interfaces which are user visible (level 5 and level 4) are considered as subject to standardization because they are the only ones which an application program would normally access.

78-02-20

PHYSICAL CONTROL (level 1)

Functionality:

To provide the physical (electrical, electromagnetic, optical, acoustical, etc.), functional and mechanical characteristics that are necessary to support the establishment, maintenance, and disconnection of a physical call between Data Terminal Equipments (DTE) via Data Circuit-terminating Equipments (DCE) and communication channel(s). In the case of switched circuits, this includes the use of network call establishment procedures.

78-02-20

PHYSICAL CONTROL (level 1)

Services of the Interface with the Next Higher Level:

- establishing circuit switched network calls
- accepting incoming circuit switched network calls
- controlling direction of transmission on HDX channel
- controlling maintenance loop tests
- selecting signal rates and transmission facilities
- disconnecting circuit switched network calls
- reporting errors and status

78-02-20

PHYSICAL CONTROL (level 1)

Protocol (Interface with Peer Level):

The set of rules by which a circuit switched call is established, maintained and terminated and data transferred. It includes the format by which control information is passed and the rules by which it is interpreted in order to transfer data via a call.

78-02-20

PHYSICAL CONTROL (level 1)

Standardization Organizations:

- U.S.A. — EIA-TR30,
  ANSI/X3S3
- International — CCITT SG VII and SG XVII,
  ISO/TC97/SC6/WG2 and WG3

78-02-20

PHYSICAL CONTROL (level 1)

Examples of existing standards:

- EIA RS-366 — Interface between DTE and automatic calling equipment for data communications
- EIA RS-422/423 — Electrical Characteristics of Balanced/Unbalanced Voltage Digital Interface Circuits
- CCITT V.28 — Electrical Characteristics for Unbalanced Double Current Interchange Circuit
- CCITT X.21 — General Purpose Interface for Synchronous Operation

78-02-20

LINK CONTROL (level 2)

Functionality:

To provide for the reliable interchange of data between Data Terminal Equipments (DTE) via a communication link.

78-02-20

LINK CONTROL (level 2)

Objects of Interest:

ET21 link-node: A set of one or more colocated logical-link-terminals.

ET22 logical-link-terminal: A point on a link in a communication network at which data can either enter or leave the link.

ET23 logical-data-link: The means of connecting one logical-link-terminal to another on the same physical link for the purpose of transmitting or receiving data.

NOTE: "ET" means an Entity Type, the first digit designated the level within the model, the second digit designated a concept defined on page 0-16 or 0-17.

78-02-20

LINK CONTROL (level 2)

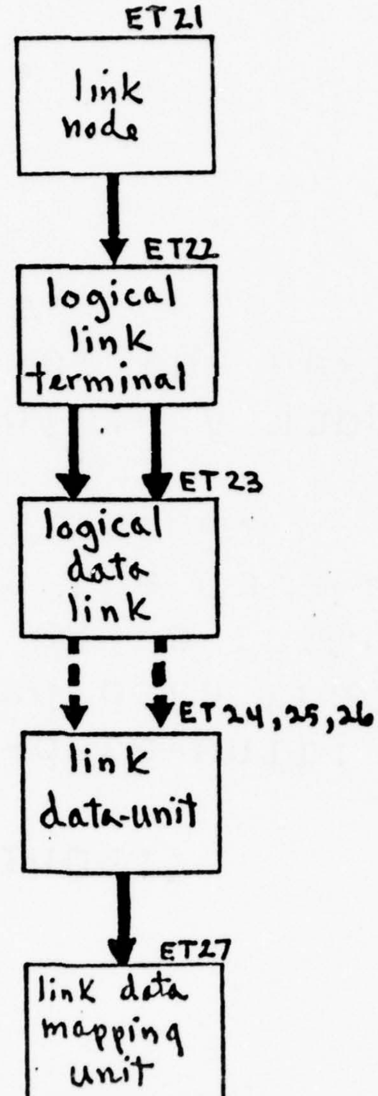
Objects of Interst (continued)

ET24, ET25 and ET26 link-data-unit: The unit of control information and data transferred over a logical-data-link, eg. a frame in HDLC. This unit also serves as the data-assurance-unit and data-flow-unit of level 2.

ET27 link-data-mapping-unit: A mapping entity used to map a data-unit of the next higher level onto a link-data-unit.

LINK CONTROL (level 2)

Relationships Between Objects:



78-02-20

LINK CONTROL (level 2)

Services of the Interface with the Next Higher Level:

- establishing, maintaining, and terminating logical-data-links
- addressing destination logical-link-terminal(s) on a logical-data-link
- transferring link-data-units
- detecting and correcting transmission errors
- controlling flow across a logical-data-link
- testing and initializing logical-data-links
- reporting errors and status

78-02-20

LINK CONTROL (level 2)

Protocol (Interface with Peer Level):

The set of rules by which a logical-data-link is established, maintained and terminated and data is transferred across a link. It includes the format by which control information is passed and the rules by which it is interpreted in order to transfer data across a logical-data-link.

78-02-20

LINK CONTROL (level 2)

Standardization Organizations:

•	U.S.A.	ANSI/X3S34
•	International	ISO/TC97/SC6/WG 1,
		CCITT SG VII,
		ECMA TC 9

78-02-20

LINK CONTROL (level 2)

Examples of Existing Standards:

- ISO-HDLC (High Level Data Link Control)
  - IS 3309 Frame Structure
  - DIS 4335 Elements of Procedure
  - DP 6159 Unbalanced Class
  - DP 6256 Balanced Class
- ANSI-ADCCP (Advanced Data Communication Control Procedure)
  - BSR X3.66
- CCITT X.25 LAP and LAP B
- U.S.A. Government Federal Standard 1003

78-02-20

TRANSPORT NETWORK CONTROL (level 3)

Functionality:

To provide for the reliable transfer of data between endpoints across a communication network, to provide interconnection services with a packet switched network, and to provide control over a private packet switched network.

78-02-20

TRANSPORT NETWORK CONTROL (level 3)

Objects of Interest:

ET31 transport-network-node: A set of one or more colocated transport-network-endpoints.

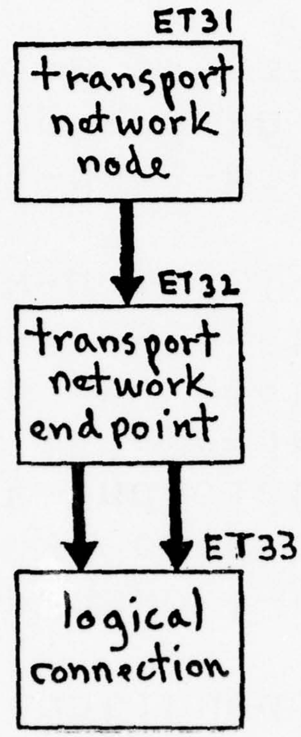
ET32 transport-network-endpoint: The point in a communication network that is the source and/or sink of data being transferred.

ET33 logical-connection: An association between two transport-network-endpoints which supports the transfer of data.

78-02-20

TRANSPORT NETWORK CONTROL (level 3)

Relationships Between Objects:



78-02-20 (data structure diagram)

TRANSPORT NETWORK CONTROL (level 3)

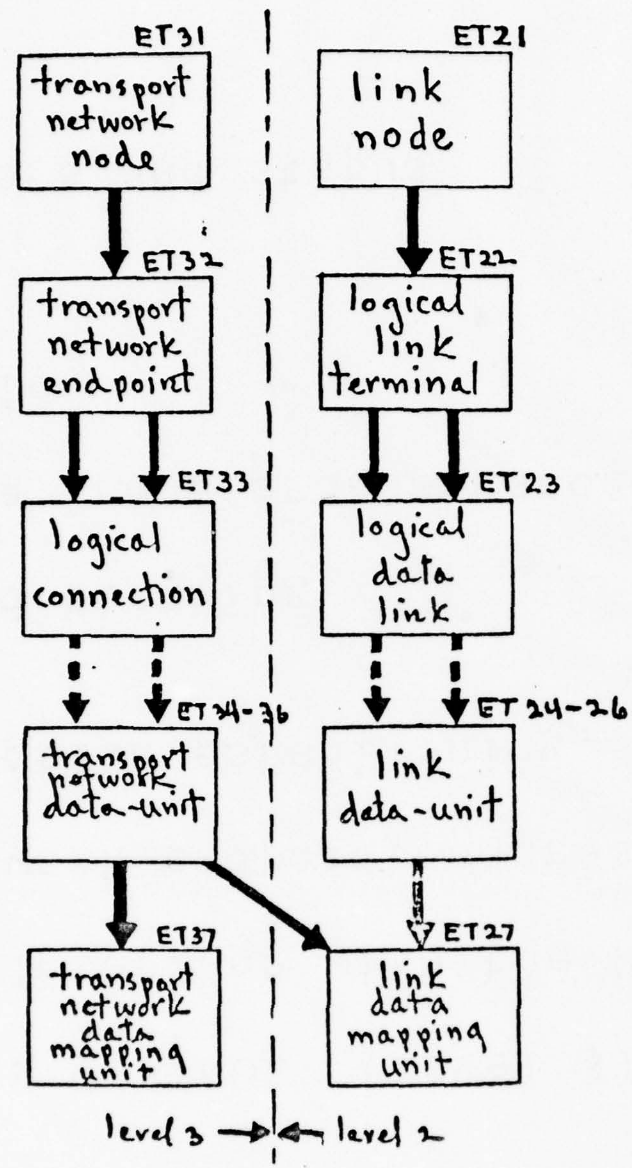
Objects of Interest (continued)

ET34, ET35 and ET36 transport-network-data-unit: A unit of data transferred between transport-network-endpoints. It has a size which is defined independently of a link-data-unit or any higher level data unit. This unit also serves as the data-assurance-unit and the data-flow-unit of level 3.

ET37 transport-network-data-mapping-unit: A mapping entity used to map a data-unit of the next higher level onto a transport-network-data-unit.

TRANSPORT NETWORK CONTROL (level 3)

Relationships Between Objects of Levels 2 and 3:



78-02-20

TRANSPORT NETWORK CONTROL (level 3)

Services of the Interface with the Next Higher Level:

- transport-network-endpoint addressing
- logical-connection establishing, maintaining and terminating
- segmenting and blocking
- providing data transfer assurance
- controlling flow
- routing
- reporting errors and status

78-02-20

TRANSPORT NETWORK CONTROL (level 3)

Protocol (Interface with Peer Level):

The set of rules by which (1) a logical-connection is established, maintained and terminated, (2) data is transferred and (3) by which traffic and routing information is transferred. It includes the format by which control information is passed and the rules by which it is interpreted in order to transfer data across a logical-connection.

Separate protocols may exist for the the different services which support this functionality, eg. virtual call service, and datagram service.

TRANSPORT NETWORK CONTROL (level 3)

Standardization Organizations:

- U.S.A. ANSI/X3S33 and X3S37
- International CCITT SG VII,
  ISO/TC97/SC6/WG 1 and WG 2

78-02-20

TRANSPORT NETWORK CONTROL (level 3)

Examples of Standard Protocols:

- ANSI/X3.57 Structure for formatting message headings for information interchange using American National Standard Code for Information Interchange for data communication systems control
- ANSI/X3 Project 281, Code Independent heading formats
- ISO/TC97/SC6 Project 24 Standardization of information field format and function for communication purposes
- CCITT X.25 Level 3; X.7x

78-02-20

SESSION CONTROL (level 4)

Functionality:

To provide for the reliable transfer of data between the Presentation Control modules supporting two processes in a network of cooperating workstations and to provide the supporting integrity and security control facilities.

78-02-20

SESSION CONTROL (level 4)

Objects of Interest:

ET41.1 session-node: A set of colocated workstations, mailboxes and transport-network-endpoints.

ET41.2 workstation: An entity representing zero, one or more processes, all having access to the same set of procedures and databases.

ET41.3 mailbox: An addressable entity assigned to a specific workstation and capable of queueing session requests until they are accepted by the recipient process. A workstation may have several mailboxes. A mailbox serves only one workstation at a time, but may be reassigned.

NOTE: There are four entity types which support the "location" concept at level 4. Three are on this page and one is on the next page.

78-02-20

SESSION CONTROL (level 4)

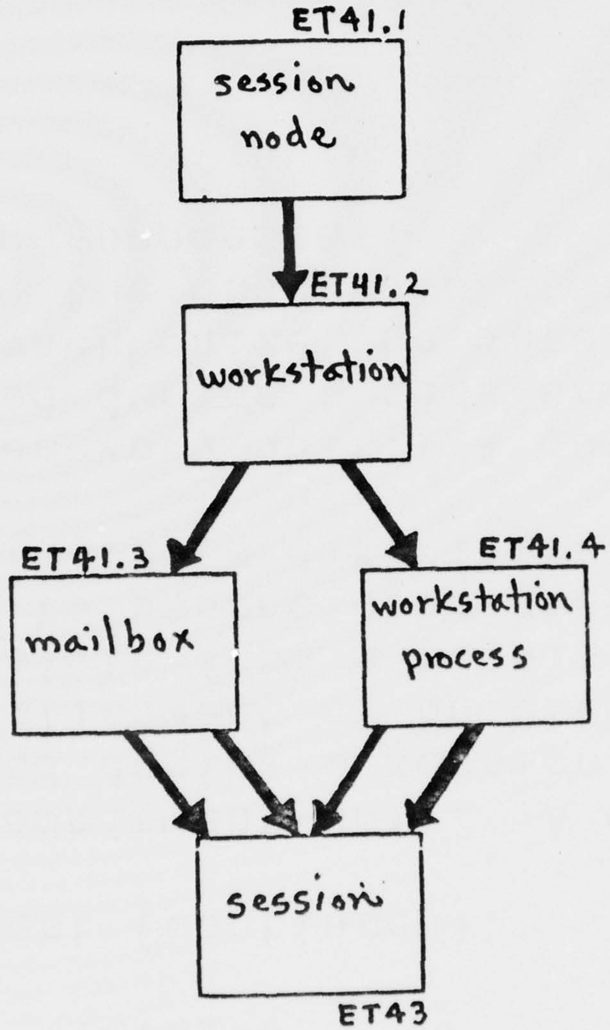
Objects of Interest (continued)

ET41.4 workstation-process: A systematic sequence of operations in execution within a workstation to produce a specified result, independently, or in cooperation with processes of the same or other workstations.

ET43 session: An association between two workstation processes to support the transfer of data between them.

SESSION CONTROL (level 4)

Relationships Between Objects:



SESSION CONTROL (level4)

Objects of Interest (continued)

ET44.1 session-commitment-unit: A quantity of data transferred during a session, used to assure the accomplishment of an element of work which is indivisible from a consistency point of view, eg. to assure consistency of database updating across a distributed systems.

ET44.2 session-recovery-unit: A quantity of data transferred during a session-commitment-unit, used to synchronize the date transferred with respect to the checkpoints of the cooperating workstation processes.

SESSION CONTROL (level 4)

Objects of Interest (continued)

ET45.1 session-interaction-unit: A subdivision of a session-commitment-unit delimited by the passing of control of a session from one workstation process to its correspondent process. The process in control of the session may request such services as the termination of sessions, session-commitment-units, session- recovery-units, or session-interaction-units. For two-way alternate sessions, only the workstation process having control may send data.

ET45.2 session-quarantine-unit: A quantity of text which cannot be released to the recipient process until the sending process signals its completion.

SESSION CONTROL (level 4)

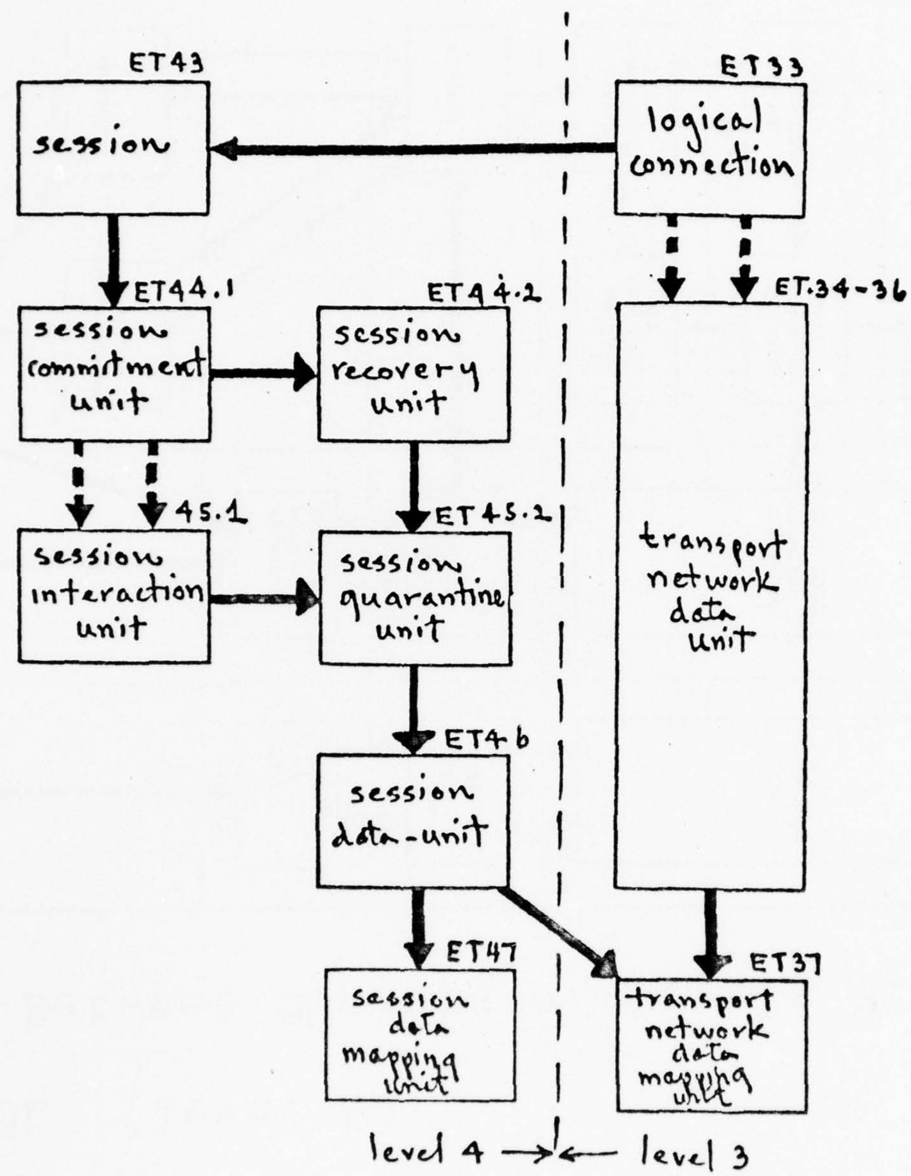
Objects of Interest (continued)

ET46 session-data-unit: A unit of data transferred between a pair of processes using a session. It has a size which is defined independently of the transport-network-data-unit size or the data-unit size of the next higher level.

ET47 session-data-mapping-unit: A mapping entity used to map the data-unit of the next higher level onto a session-data-unit.

SESSION CONTROL (level 4)

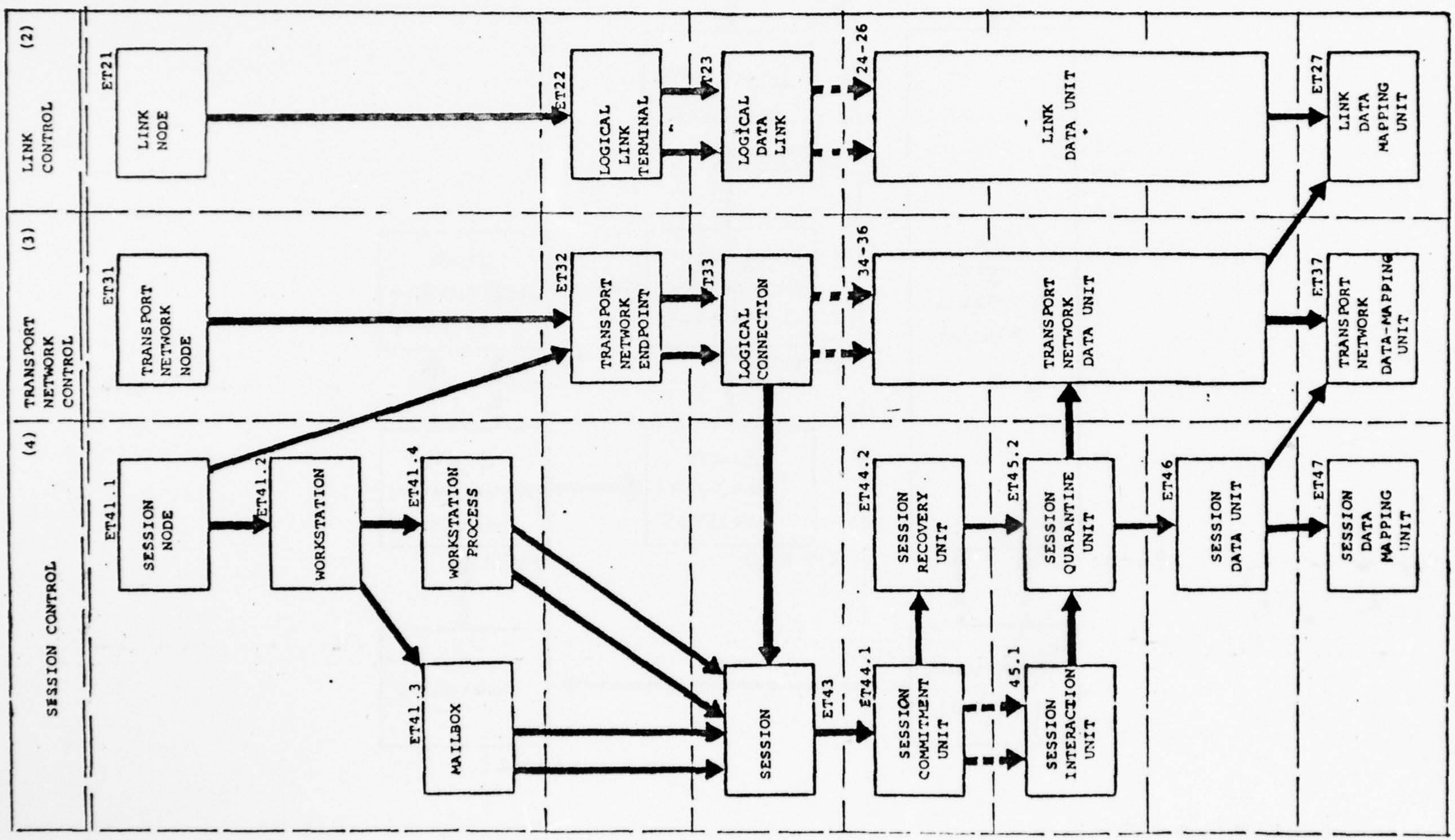
Relationships Between Objects of Levels 3 and 4:



78-02-20

SESSION CONTROL (level 4)

Relationships Between Objects of Levels 2, 3 and 4:



78-02-20 (data structure diagram)

SESSION CONTROL (level 4)

Services of the Interface with the Next Higher Level:

- addressing workstations via mailboxes
- session establishing, maintaining and terminating including requesting process creation when specified
- sending data and delimiting data enclosures
- notifying processes upon receipt of data and data enclosure delimiters
- segmenting and blocking
- buffering data and controlling flow
- quarantining
- supporting checkpoint, recovery and commitment

78-02-20

SESSION CONTROL (level 4)

Protocol (Interface with Peer Level):

The set of rules by which a session is establishied, maintained and terminated and data transferred. It includes the format by which control information is passed and the rules by which it is interpreted in order to transfer data across a session.

78-02-20

SESSION CONTROL (level 4)

Standardization Organizations:

- U.S.A. ANSI/X3/SPARC/Study Group - Distributed Systems
- International ISO/TC97/SC16, ISO/TC97/SC6

78-02-20

SESSION CONTROL (level 4)

Examples of Existing Standards:

• U.S.A.	FIPS (User-Terminal Protocols - Entry and Exit Procedures Between Terminal Users and Computer Services)
• International	(none known)

78-02-20

PRESENTATION CONTROL (level 5)

Functionality:

To adapt existing and specialized information handling characteristics of a process to the Session Control interface, including transforming commands and data as necessary to accomodate differences in sending and receiving formats, data types, data codes, and data representations.

78-02-20

PRESENTATION CONTROL (level 5)

It is anticipated that there will be a number of alternate forms of Presentation Control, each designed to handle a particular set of information handling characteristics. Information handling characteristics which will require support include the following:

- COBOL Communication Task Group Interface
- COBOL File I/O Interface
- FORTRAN File I/O Interface

PRESENTATION CONTROL (level 5)

Objects of Interest:

COBOL Concepts (level 5)		Session Concepts (level 4)
symbolic-source	-----------	mailbox (source)
symbolic-destination	------	mailbox (destination)
queue		
subqueue-1		
subqueue-2		
subqueue-3	----------------	session
		session-commitment-unit
		session-recovery-unit
message-group	-------------	session-interaction-unit
message	-------------------	session-quarantine-unit
segment	-------------------	session-data-unit
portion	-------------------	session-data-mapping-unit

78-02-20

PRESENTATION CONTROL (level 5)

Services of the Interface with the Next Higher Level:

- encrypting/decrypting
- compacting/expanding
- character transforming, eg. ASCII, EBCDIC, BCD
- network virtual terminal to/from display format transforming, eg. X3L2 work on 8 bit ASCII
- record format to/from display format transforming
- data format, code and representation transforming
- command transforming, eg. COBOL CTG to Level 4 Interface

78-02-20

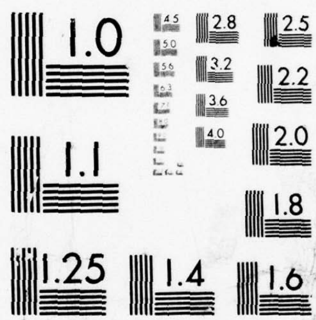
AD-A068 267 PURDUE UNIV LAFAYETTE IND PURDUE LAB FOR APPLIED IND--ETC F/G 9/2
MINUTES 1978 SPRING REGIONAL MEETING INTERNATIONAL PURDUE WORKS--ETC(U)
JUN 78 N00014-78-C-0127

UNCLASSIFIED NL

4 OF 4

AD
A068267

END
DATE
FILMED
6 -- 79
DDC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

PRESENTATION CONTROL (level 5)

Examples of Standardization Organizations:

• U.S.A.	ANSI/X3/J1 (PL/I)
	ANSI/X3/J2 (BASIC)
	ANSI/X3/J3 (FORTRAN)
	ANSI/X3/J4 (COBOL)
	ANSI/X3/L2 (Codes & Character Sets)
• International	ISO/TC97/SC16 (Open Systems Interconnection)
	ISO/TC97/SC15
	ISO/TC97/SC 5 (Programming Lang.)
	ISO/TC95 (Facsimile Project)

78-02-20

PRESENTATION CONTROL (level 5)

Examples of Standards:

- U.S.A. ASCII code set
- International (none known)

78-02-20

PROCESS LEVEL (level 6)

Functionality:

To support application/system activities including the exchange information between processes of manual, computerized and industrial workstations and thus to cooperate in the achievement of the objectives of an enterprise. All lower levels are designed to facilitate this end.

PROCESS LEVEL (level 6)

Objects of Interest:

The objects of interest will be specific to each different inter-process protocol. No generalization can be made.

For a bank's savings protocol, they might be: customers, accounts, deposits and withdrawals.

For a manufacturer's order entry protocol, they might be: customer, products, production orders, sales orders, inventory locations and stock.

For a distributed database access protocol, they might be: files, records, sets, items, access requests, processes and data descriptions.

For a programming language, they might be: procedures, statements, labels, items and references.

PROCESS LEVEL (level 6)

Services of the Interface with the Next Higher Level:

NOTE: There is no higher level defined within the reference model, therefore no interface with the next higher level.

78-02-20

PROCESS LEVEL (level 6)

Protocol (Interface with Peer Level):

The inter-process protocols specify a set of rules by which the information transferred between processes of two workstations is to be formatted and interpreted. There will be many different inter-process protocols which fall into two major classes:

- information systems protocols
- application system protocols
  - industry standard
  - enterprise specific

78-02-20

PROCESS LEVEL (level 6)

Examples of Information Systems Protocols:

- remote batch processing protocol
- program development protocol
- programming languages (COBOL, FORTRAN, etc.)
- distributed database access protocol
- file transfer protocol
- integrity control protocol
- security control protocol
- workstation control protocol
- terminal interaction protocol

78-02-20

PROCESS LEVEL (level 6)

Examples of Industry Standard Application Protocols:

- electronic funds transfer protocol
- Internal Revenue Service (USA) protocol
- electronic mail protocol
- library interchange protocol
- civil aviation protocol

78-02-20

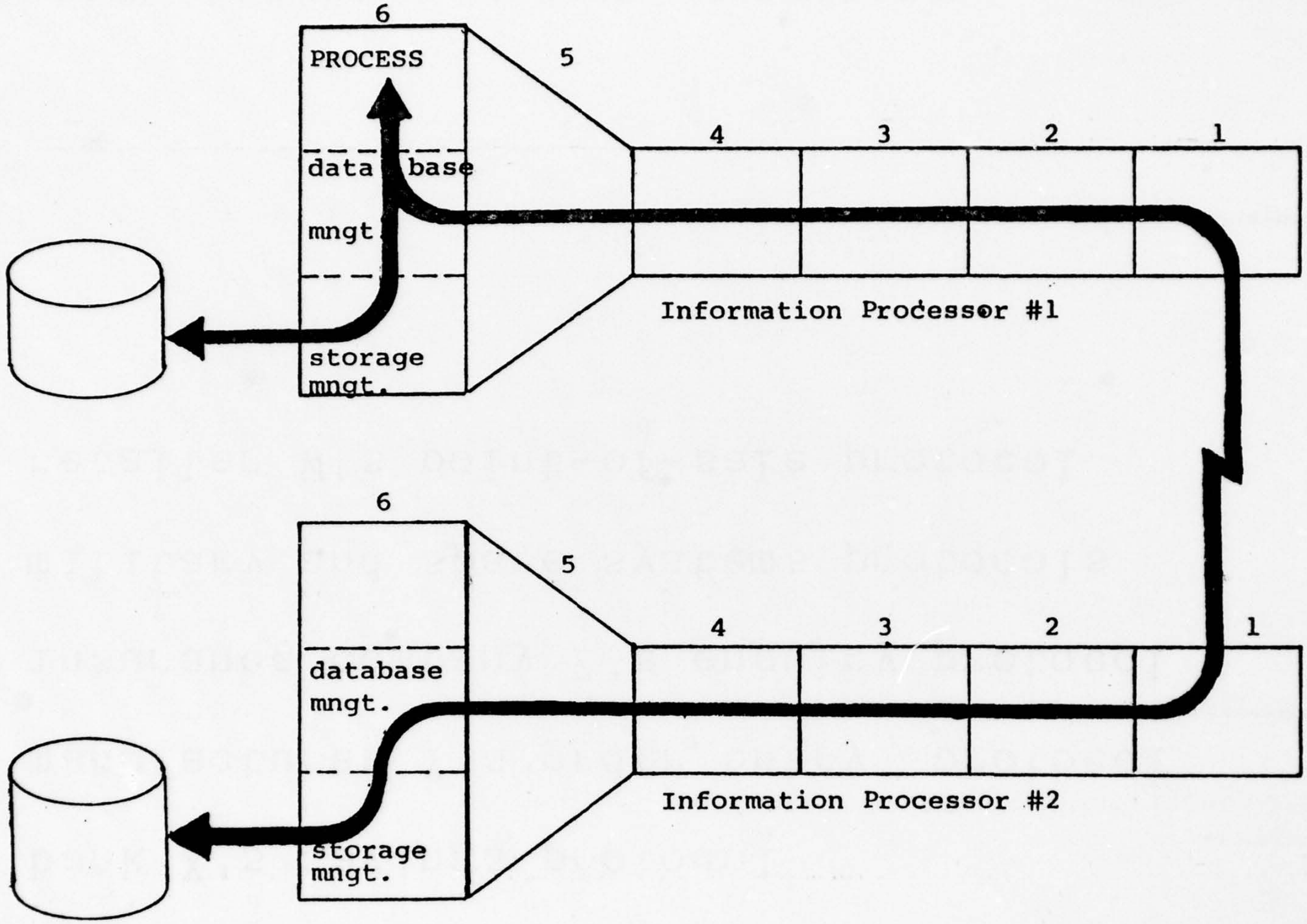
PROCESS LEVEL (level 6)

Examples of Enterprise Specific Protocols:

- bank X's savings protocol
- manufacturer Y's order entry protocol
- insurance company Z's enquiry protocol
- military and space systems protocols
- retailer W's point-of-sale protocol

78-02-20

Distributed Database Access Protocol: (level 6)



The Database Management System determines whether it can directly access the data requested or whether it requires help from a surrogate database process at another session-node.

78-02-20

PROCESS LEVEL (level 6)

Protocol (Interface with Peer Level):

Each such level 6 protocol would be defined in terms of:

- the data formats and their interpretation
- the presentation control services used
- the session control services used
- the integrity control services used
- the security control services used

78-02-20

PROCESS LEVEL (level 6)

Examples of Standardization Organizations:

- ANSI/X3J4 (COBOL)
- ANSI/X3L5 (Labels & Files)
- Transportation Data Coordination Committee
- International Civil Aviation Organization

78-02-20

PROCESS LEVEL (level 6)

Examples of Existing Standards:

- ANSI COBOL (1973)
- National Commission on Library and Information Services Protocol (USA)
- Transportation Data Coordination Committee - U.S. Industry Electronic Data Interchange (EDI) Protocol
- International Civil Aviation Organization - Automated Data Interchange System Protocol

78-02-20

HUMAN/MACHINE INTERFACE

Manual Workstations:

Manual workstations interface with computerized or industrial workstations and some other manual workstations with the assistance of terminal and unit record operators, terminals and unit record equipment, and terminal control workstations.

78-02-20

HUMAN/MACHINE INTERFACE

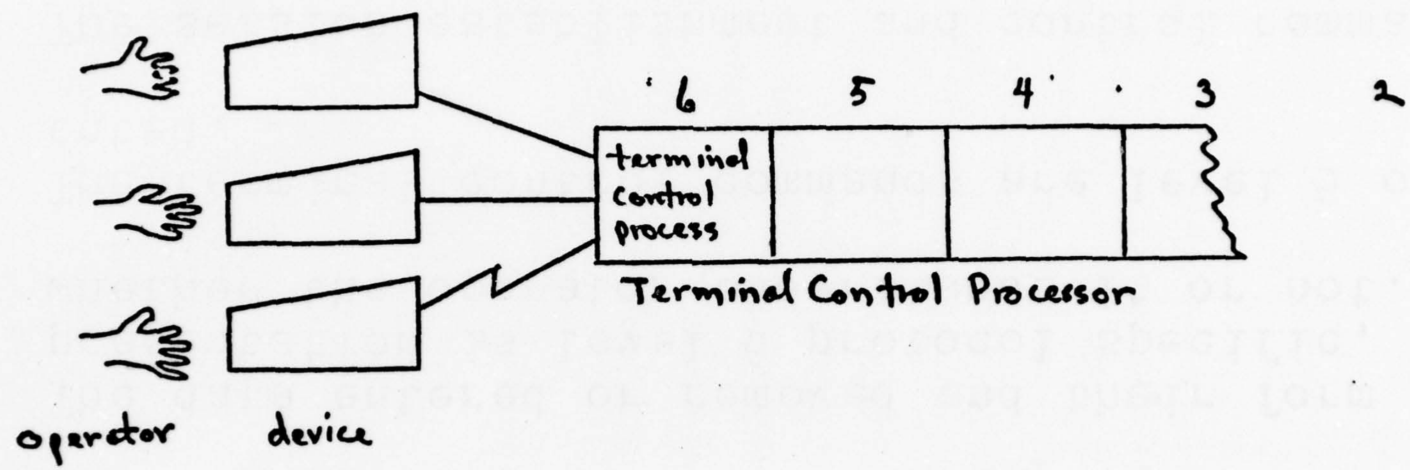
Operator:

Each terminal or unit record operator is required to operate with some knowledge concerning levels 4, 5 and 6:

- The data entered or removed and their form of presentation is level 6 protocol specific, whether the operator understands it or not.
- The terminal control commands are level 5 oriented.
- The session establishment and control commands are level 4 oriented.

HUMAN/MACHINE INTERFACE

Terminal Control Workstation:



The terminal control workstation is a special case of an computerized workstation which acts as a surrogate of a manual workstation. It supports manual processes, as represented by their operator's actions, and makes these actions appear to be the actions taken at the initiative of the terminal control workstation's own processes.

78-02-20

HUMAN/MACHINE INTERFACE

Functionality of Terminal Control Workstation:

- to control and interpret the operation of devices at the manual workstation
- to execute commands on the behalf of the process of the manual workstation
- to receive commands from the correspondent process associated with the session and pass them on to the process of the manual workstation

SUMMARY

Schedule:

- Adopt an ISO/TC97/SC16 reference model by March 1979.
- Adopt ISO/TC97 Standards for selected interfaces and protocols:
  - Transport Network protocol by August 1979.
  - Session Control interface and protocol by December 1980.
  - COBOL CTG Presentation Control services by December 1980.
  - Remote File Access Protocol by December 1980.
  - Remote Job Entry Protocol by December 1980.

78-02-20

TUTORIAL NUMBER IV

PROCESS INTERFACE COMPARISONS

Mr. Maris Graube
Tektronix
Box 500 MS 50/454
Beaverton, OR 97077

Presented at the Purdue Americas Meeting

Maris Graube
Tektronix

Summary of the talk given at Purdue International Workshop
April 11, 1978.

Up to this point the interface committee has come up with functional requirements for the process control interface. Now it is going into a phase of examining what existing systems satisfy its requirements. But, even if a system satisfies the functional requirements:

a) How well does it really perform?
b) How does it compare to other systems that also satisfy the requirements?
c) What are the weaknesses of a system?

We need some criteria for comparing interfaces.

To compare all levels of protocols of all interfaces is too big a job. Instead, only the physical link will be examined here. The physical link is important. From the experience of the IEEE 488 interface, we know that a lot of problems can be avoided by paying more attention to the basics. Without a good physical link the rest of the interface system is compromised. Also, there is "loose coupling" between the physical link and the rest of the protocols so that the physical link can be examined somewhat independently. Thus, the orientation of the talk is horizontal, not vertical (Slide 1).

The physical link considerations fall into two major categories, Medias and Modulations, plus some associated consideration. (Slide 2). Examining all the existing physical links is too big a task. Therefore, those links that clearly do not satisfy our functional requirements are not examined here. The functional requirements applicable to the physical link are shown in Slide 3.

The systems specifically not considered here are loop networks and optical links because they store and forward data and because the subsystems in the network are not operationally isolated. (Slide 4).

Once optical links have been eliminated from comparison, the only remaining medias are twisted pair and coaxial cable. Slide 5 shows the comparison of some representative types.

The media comparison chart gives only a rough indication of cable characteristics. To evaluate any real system one needs to examine the real thing. (Slide 6). This is done by running pseudo random data through the cable driver and down a loaded cable and then examining the resulting "eye" pattern on the

far end of the cable. The bigger the eye pattern the better the data can be recovered. The eye pattern in turn depends on how the data is modulated or called by the cable driver.

Modulations fall into two basic categories: baseband and carrier. (Slide 7). Slides 8 and 9 show some representative types and their frequency spectra. The criteria for comparing baseband modulations is shown in Slide 10. Slide 11 shows the three basic forms of carrier modulations. Slide 12 shows the relative noise immunities of carrier modulations and baseband modulation. Slide 13 lists some of the advantages and disadvantages of carrier modulations.

Slide 14 lists some of the considerations associated with the physical link. It is these considerations that will determine the characteristics of the physical link:

a) Intrinsic safety is determined by using an apparatus like the one shown in Slide 15. Various circuits have been analyzed and the results are shown in diagrams such as the one in Slide 16. Unfortunately, intrinsic safety for cable has not been determined. Thus, we cannot yet specify the signal power on our interface cable. The maximum signal power determines the length of the interface cable, the noise immunity, the number of stations, etc.

b) There are two main ways to distribute the interface cable, shown in Slide 17. The main cable can go to each station directly or coupler boxes can tap into the main cable and auxiliary lines feed the stations. The latter method can significantly increase the effective length of the physical link.

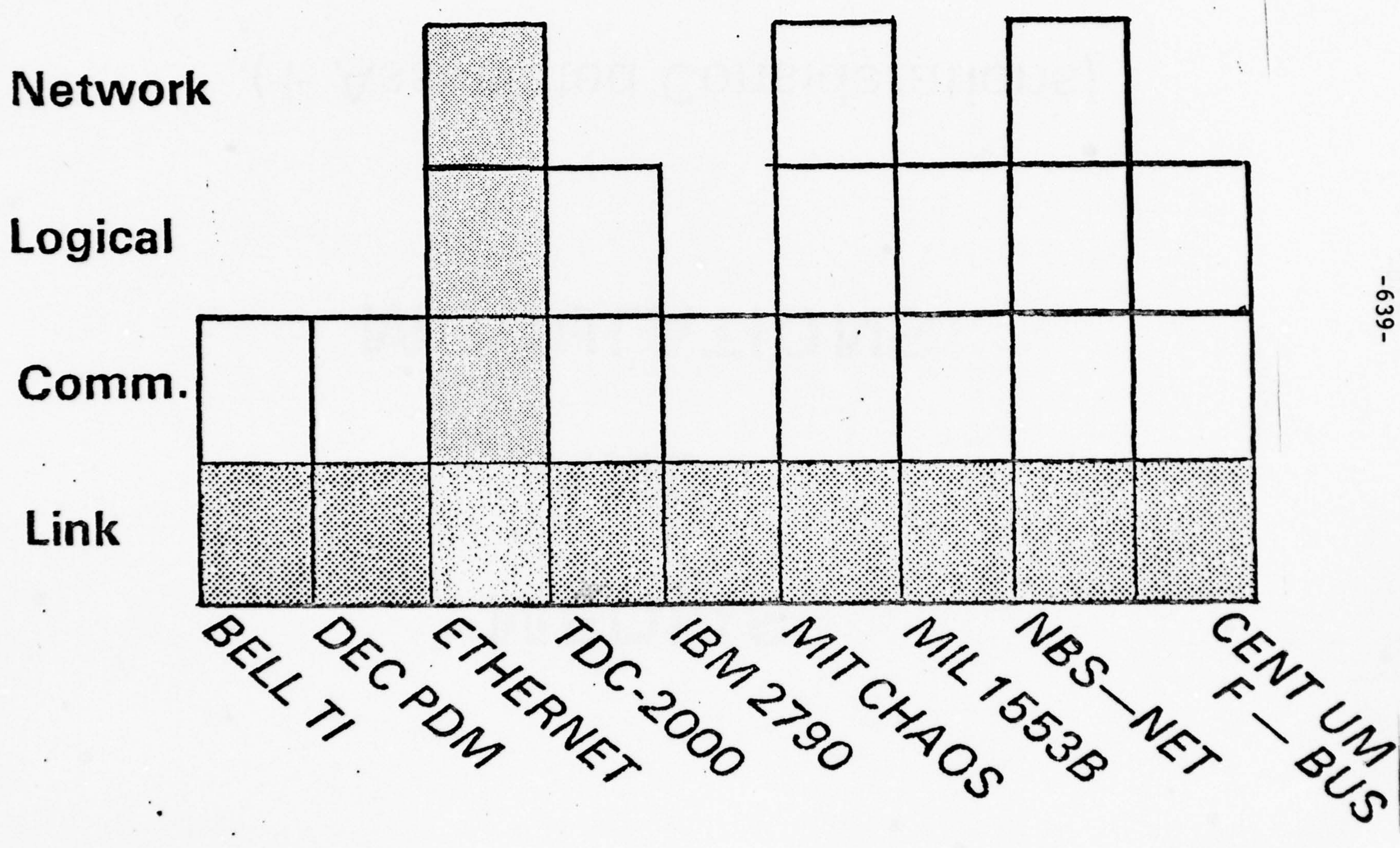
c) The requirements of tapping and coupling the cable are listed on Slide 18. These requirements can be implemented in a variety of ways. Two of them are shown in Slide 19.

c) Noise immunity is one of our functional requirements but so far there is very little information on how we can satisfy this requirement. (Slide 20). Like intrinsic safety, noise immunity is a very basic consideration on which other physical link characteristics depend such as length, data rate, etc.

Slide 21 lists a number of existing systems and shows some of their characteristics. Note that none of the systems have any rating as to intrinsic safety and that only one of them has anything in the noise category.

I bleieve that both the intrinsic safety and the noise characteristics are very basic considerations. These mundane characteristics should be well established by our committee before we go on to the more exciting physical link considerations.

System-Protocol Matrix



MEDIAS

AND

MODULATIONS

(+ Associated Considerations)

Physical Link Functional Requirements

3.4 2K METRES; 1 MBIT/SEC.

4.2 DIRECT DATA INTERCHANGE BETWEEN SUBSYSTEMS: NO STORE-AND-FORWARD

6.2 ISOLATED SUBSYSTEM; SYSTEM NOT DISTURBED BY POWER ON/OFF, ETC.

8.0 — MANDATORY STANDARDS.
— LIGHTNING PROTECTION.
— INTRINSIC SAFETY.

9.1 OPERATION IN ELECTRICALLY NOISY ENVIRONMENTS.

10.1 FAILURE IN STATION NOT TO CRASH WHOLE SYSTEM.

Systems Not Considered: LOOP NETWORKS—

STORE-AND-FORWARD

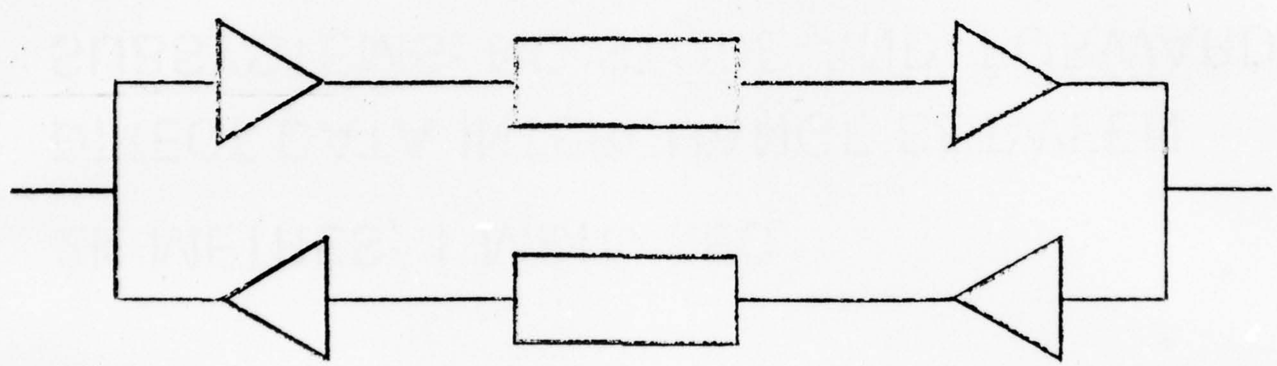
IOSLATED SUBSYSTEM?

Media Not Considered: FIBRE OPTICS—

NO PASSIVE TAPS AND SPLITTERS

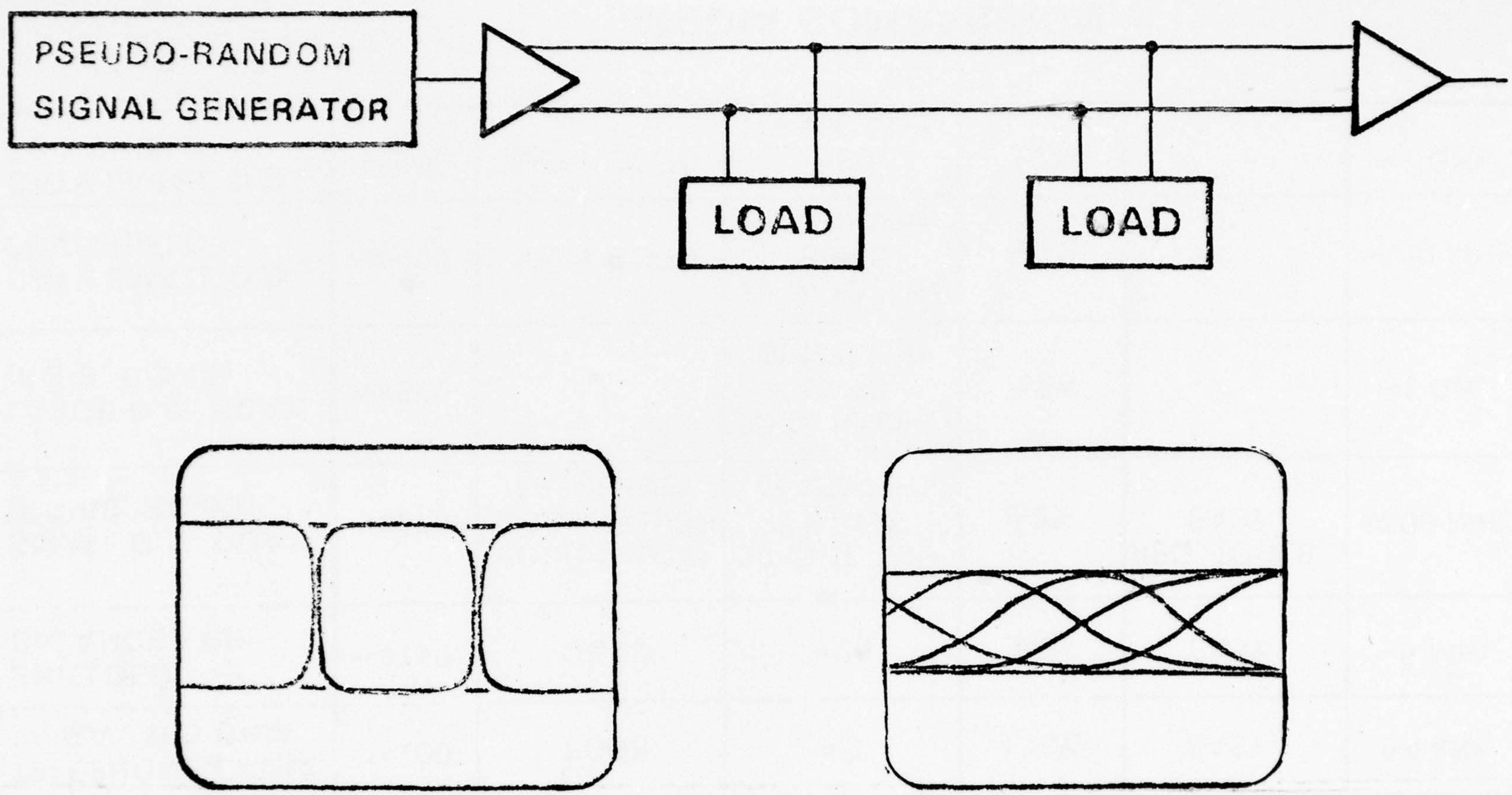
STORE-AND-FORWARD

NOT ISOLATED SUBSYSTEM



	$/KM	NOISE IMMUNITY	ATTENUATION dB/KM	ZO UNIFORMITY	EASE OF TAP	UPPER FREQUENCY
TELEPHONE CABLE 24 GA., 100 OHM	<$100	POOR	≈2	±10%	EASY	≈4 kHz
SHIELDED BALANCED PR.	≈$150	GOOD	≈20	±5%	EASY	≈5 MHz
SMALL O.D. COAX (RG-58, RG-59)	≈$250	POOR @ LOW, GOOD @ HIGH FREQUENCY	38 @ 10 MHz, 110-175 @ 100 MHz	±2%	REQ. TOOLS, EASY	≈500 MHz
LARGE O.D. COAX (RG-8, RG-11)	≈$500	"	20 @ 10 MHz, 60-70 @ 100 MHz	±2%	"	≈1 GHz
CATV SMALL O.D. FOIL SHIELD	≈$400	VERY GOOD	26 @ 10 MHz, 85-95 @ 100 MHz	±2%	"	≈500 MHz
CATV LARGE O.D. FOIL SHIELD	≈$500	VERY GOOD	≈50 @ 100 MHz	±2%	"	≈1 GHz

MEDIA COMPARISON



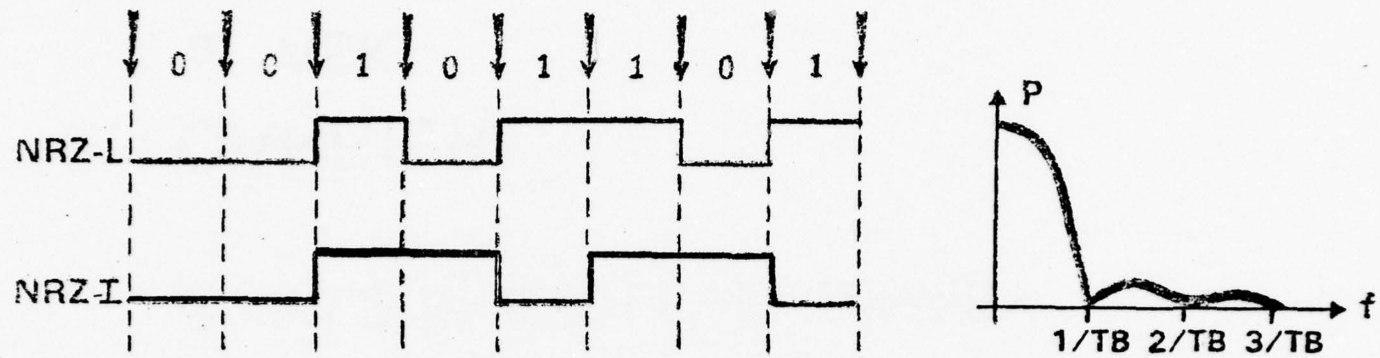
EYE PATTERN

MODULATIONS

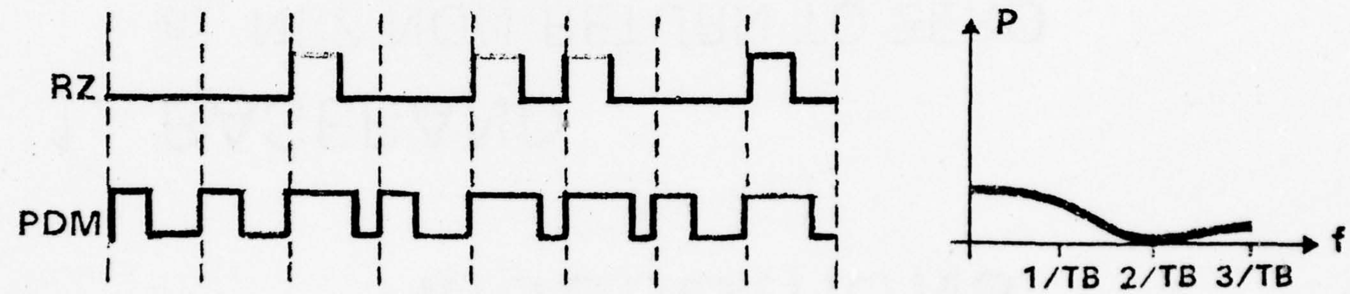
1. BASEBAND
   a. NRZ NON-RETURN TO ZERO
   b. RZ RETURN TO ZERO
   c. PE PHASE ENCODED
   d. MLB MULTI-LEVEL BINARY

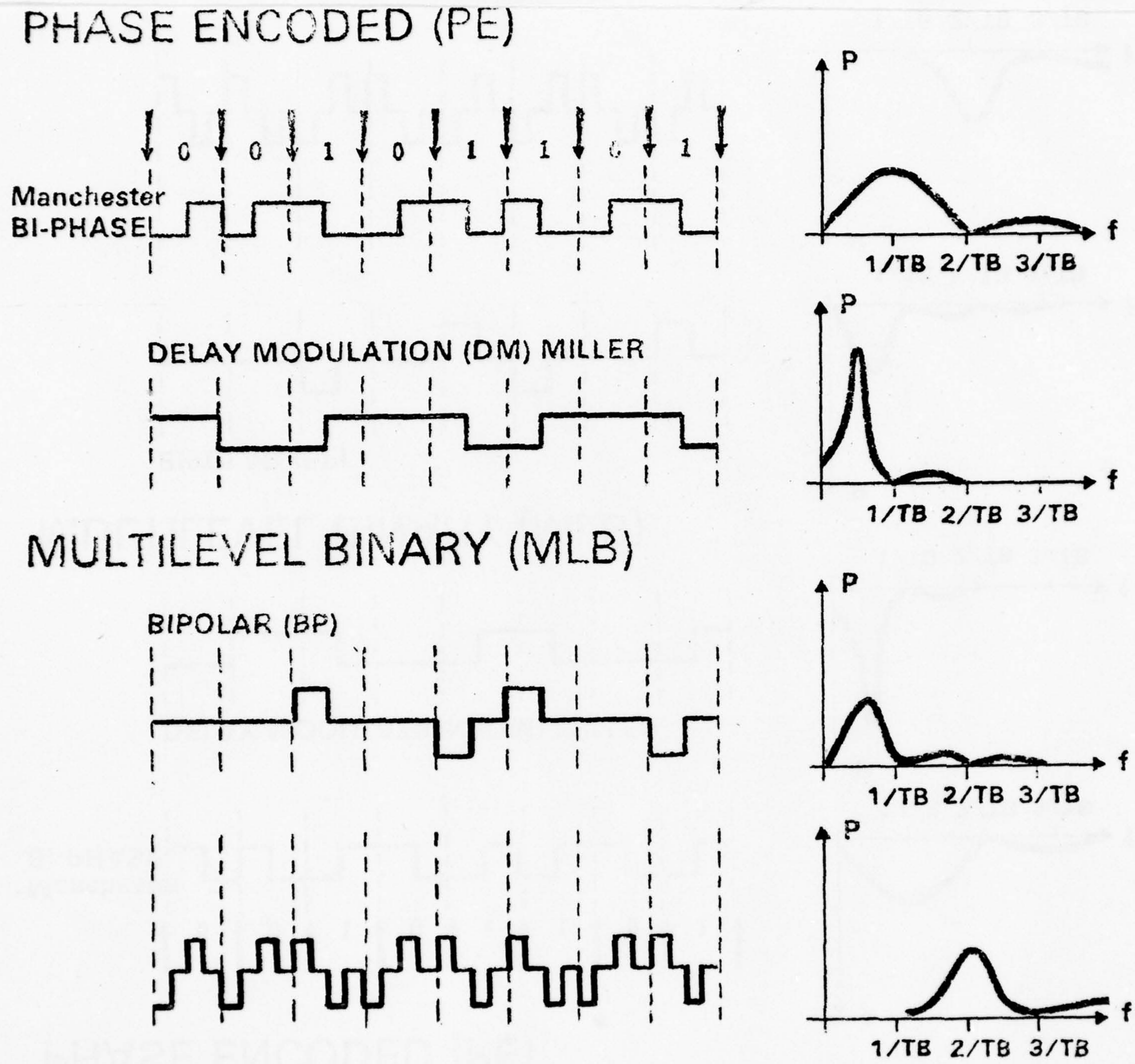
2. CARRIER
   a. ASK
   b. FSK
   c. PSK

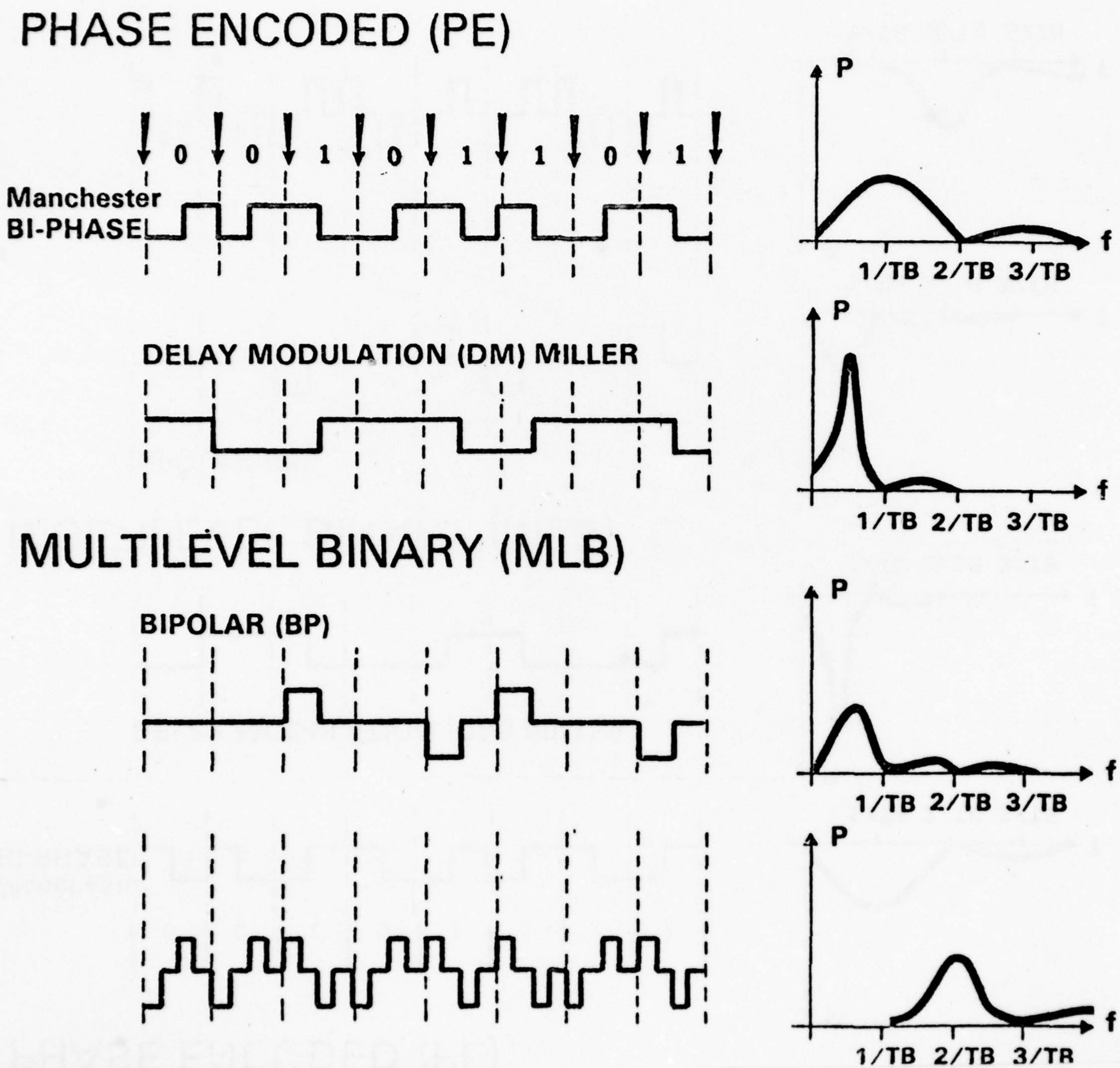
NON-RETURN TO ZERO (NRZ)



RETURN TO ZERO (RZ)



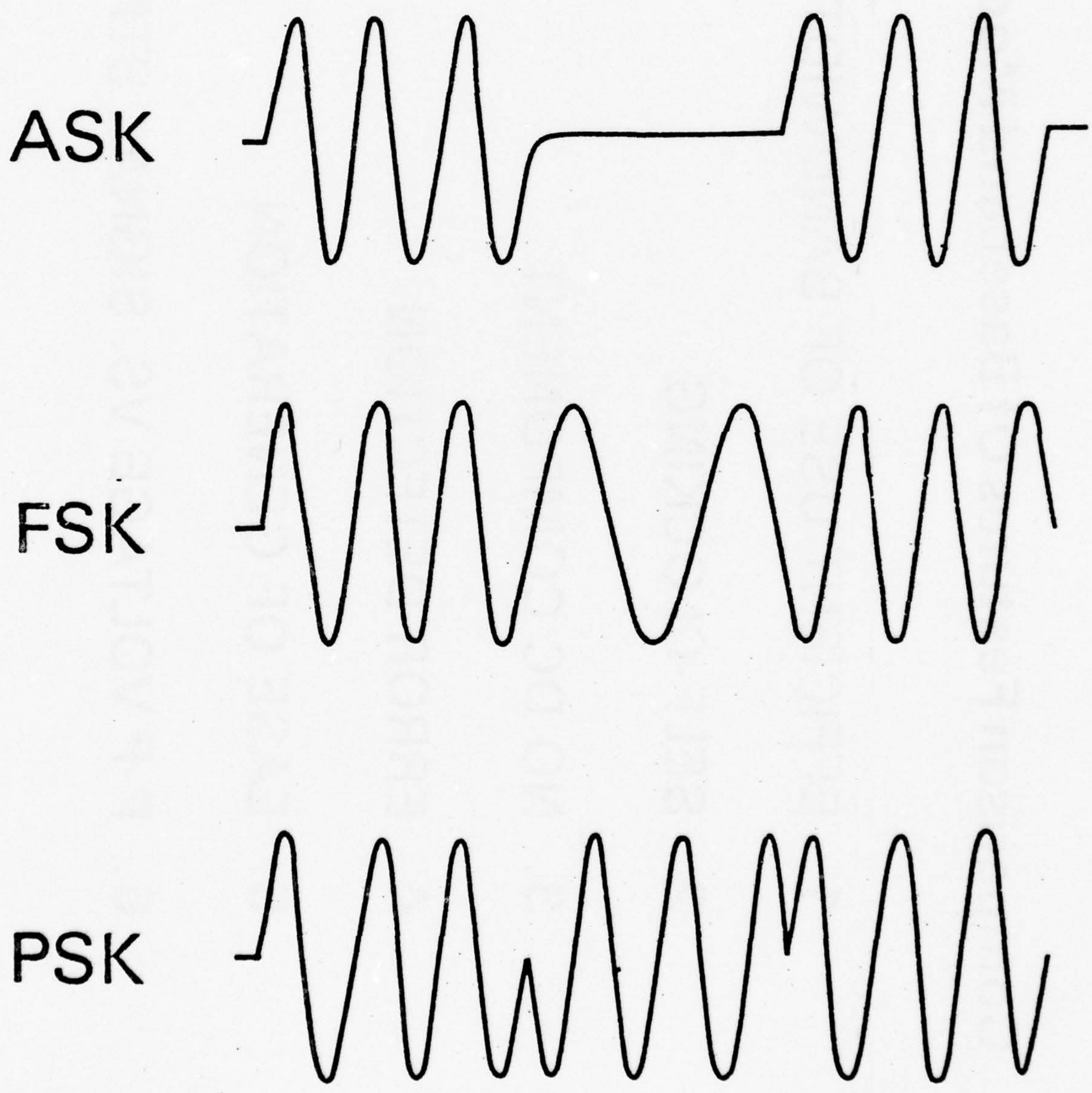

PHASE ENCODED (PE)
0 0 1 0 1 1 0 1
Manchester
BI-PHASE
DELAY MODULATION (DM) MILLER
MULTILEVEL BINARY (MLB)
BIPOLAR (BP)
P
f
1/TB 2/TB 3/TB

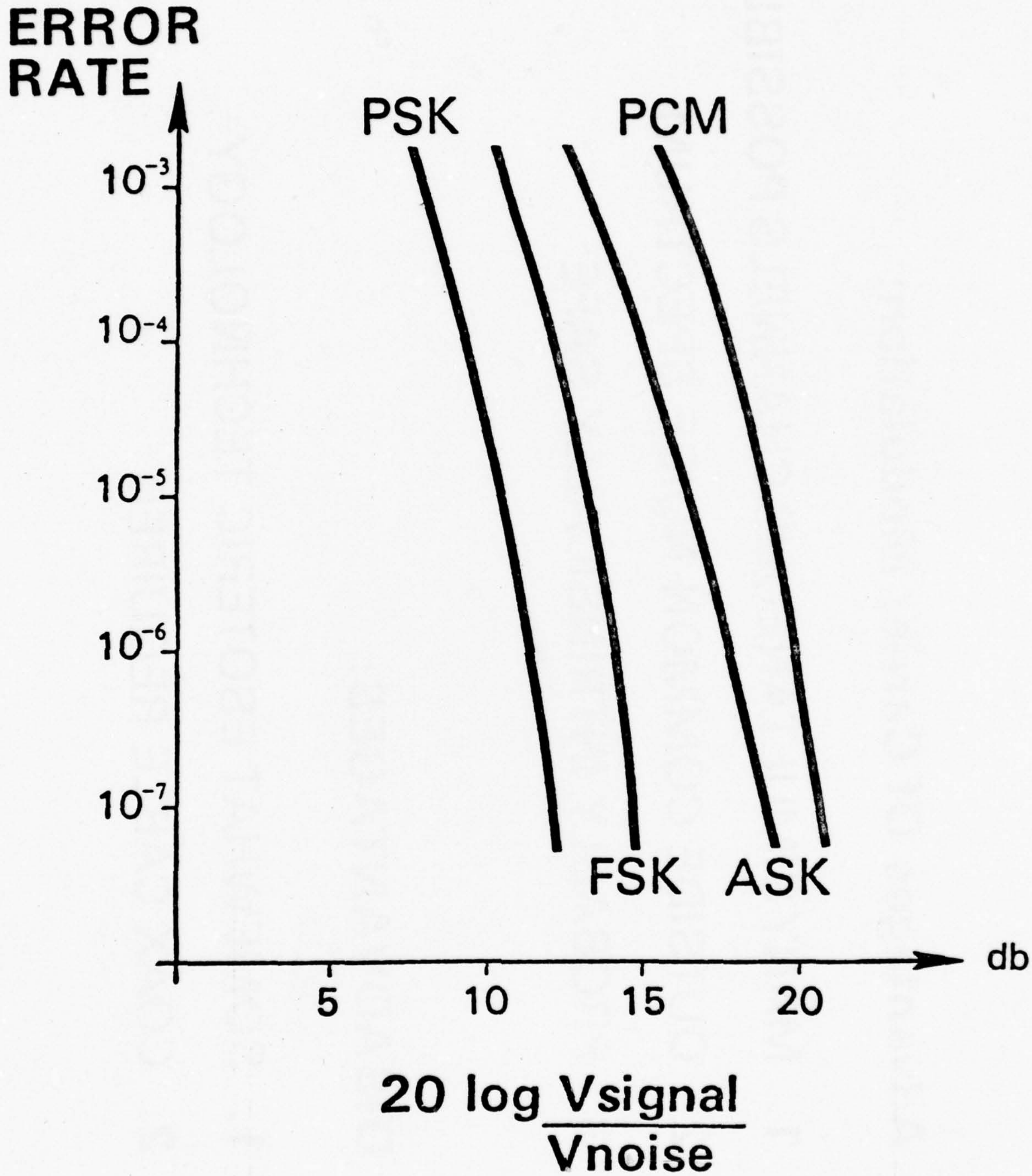

PHASE ENCODED (PE)
0 0 1 0 1 1 0 1
Manchester
BI-PHASE
P
f
1/TB 2/TB 3/TB
DELAY MODULATION (DM) MILLER
P
f
1/TB 2/TB 3/TB
MULTILEVEL BINARY (MLB)
BIPOLAR (BP)
P
f
1/TB 2/TB 3/TB
P
f
1/TB 2/TB 3/TR

Comparison Features Of Baseband Modulations:

1. EFFICIENT USE OF BANDWIDTH
2. SELF-CLOCKING
3. NO DC COMPONENT
4. ERROR DETECTION
5. EASE OF GENERATION
6. P-P VOLTAGE VS. SIGNAL STRENGTH.

CARRIER DATA MODULATIONS




ERROR
RATE
PSK
PCM
10^{-3}
10^{-4}
10^{-5}
10^{-6}
10^{-7}
FSK
ASK
db
5
10
15
20
20 log $\frac{Vsignal}{Vnoise}$

Advantages Of Carrier Modulation:

1. MANY SIMULTANEOUS CHANNELS POSSIBLE
2. OUTSIDE COMMON NOISE SPECTRUM
3. PROBABLY INTRINSICALLY SAFE

DISADVANTAGES:

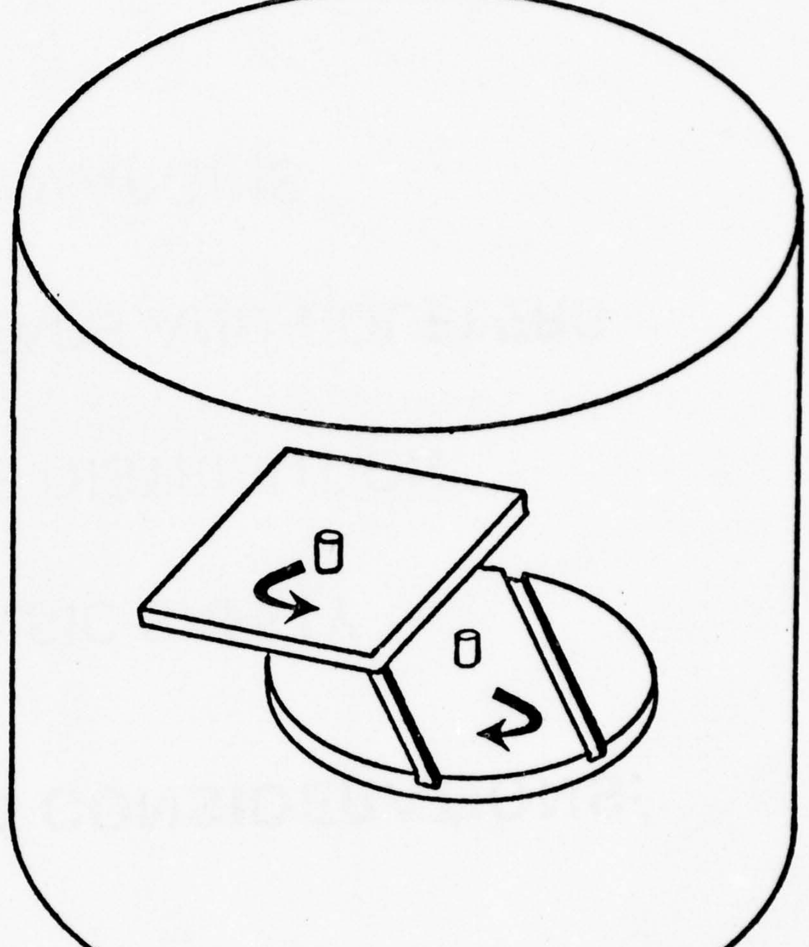
1. SOMEWHAT ESOTERIC TECHNOLOGY
2. COAX CABLE REQUIRED

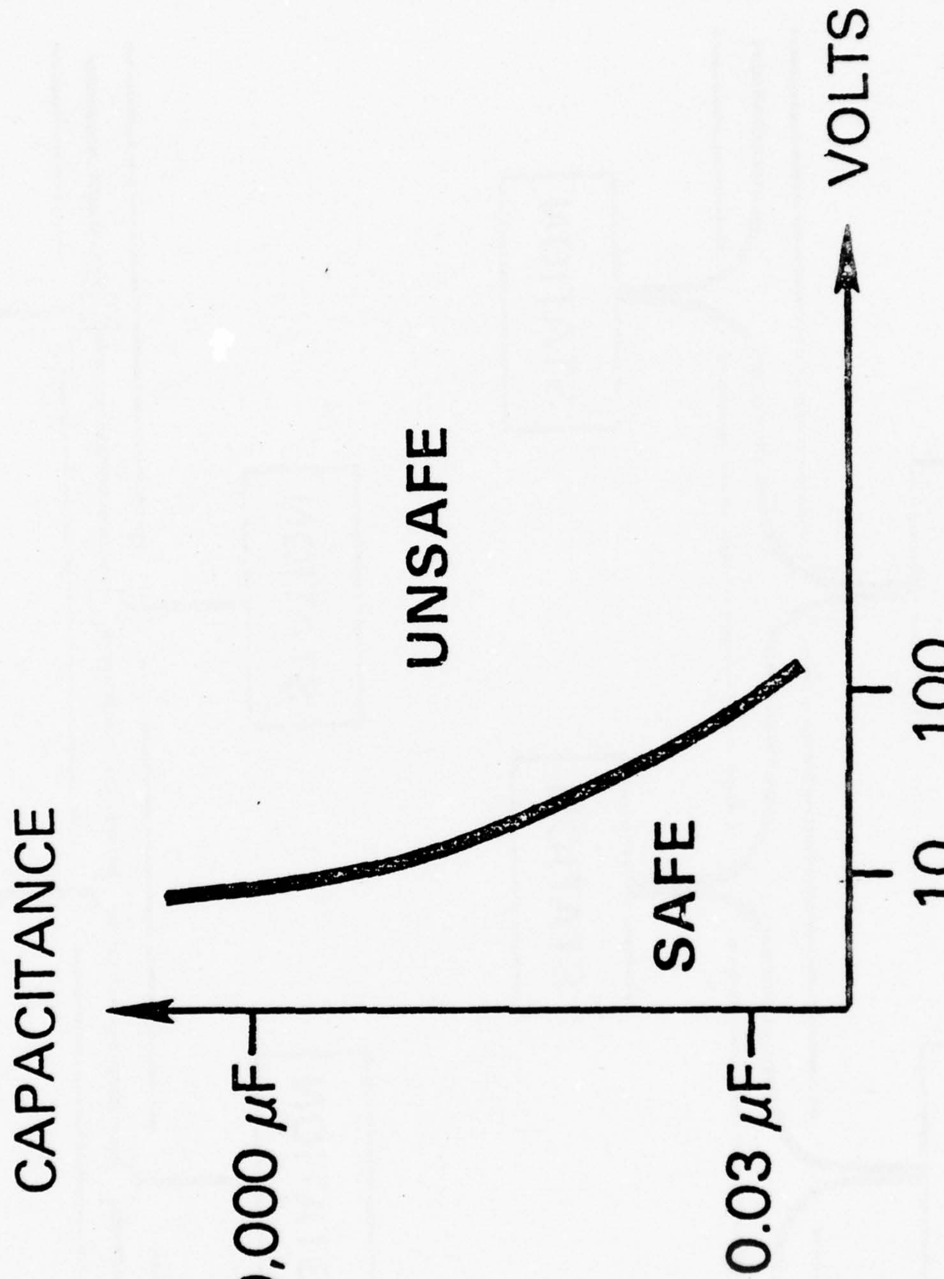
ASSOCIATED CONSIDERATIONS:

1. INTRINSIC SAFETY
2. CABLE DISTRIBUTION
3. LINE TAPS AND COUPLERS
4. NOISE MARGINS

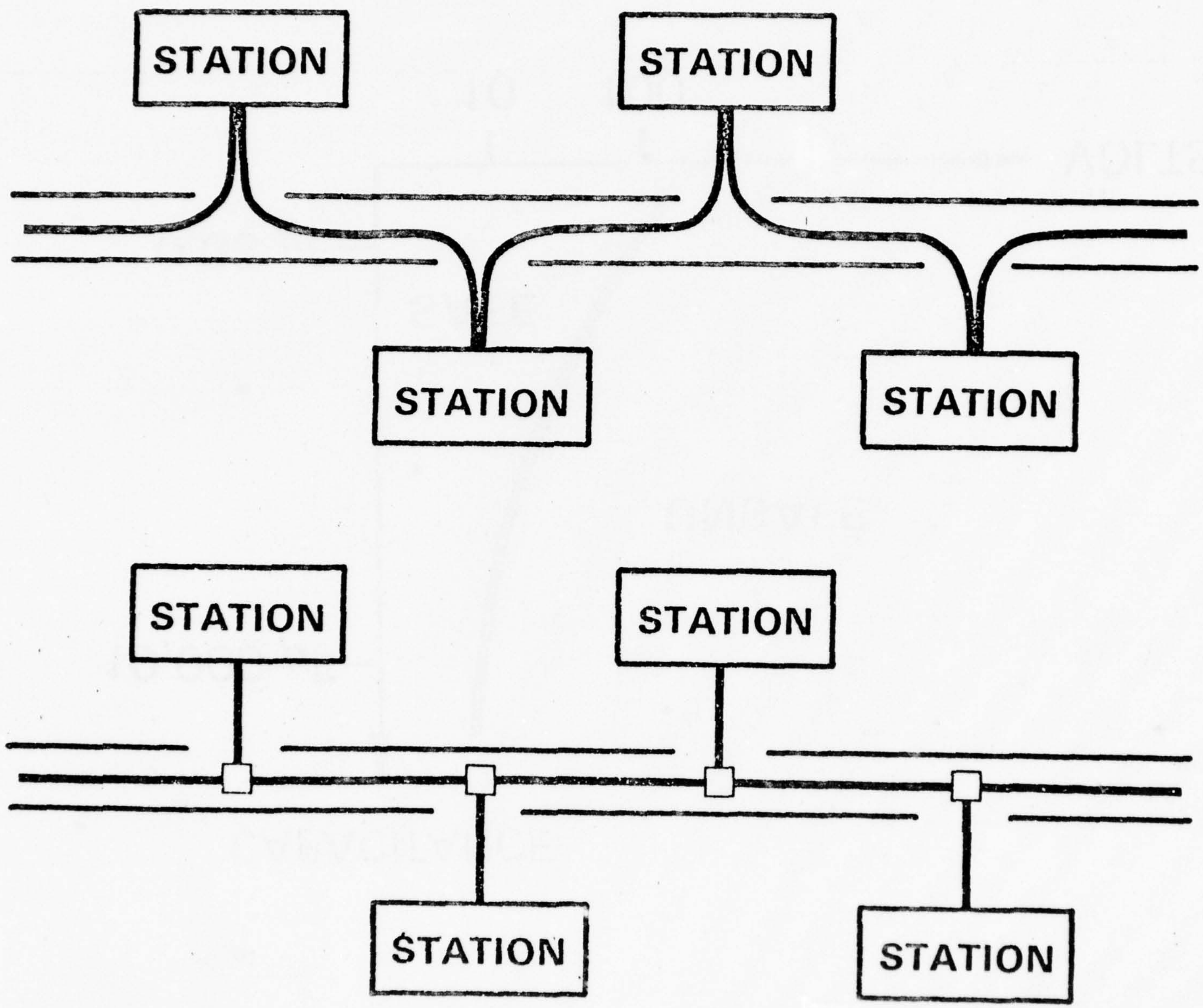
SPARK TEST

APPARATUS



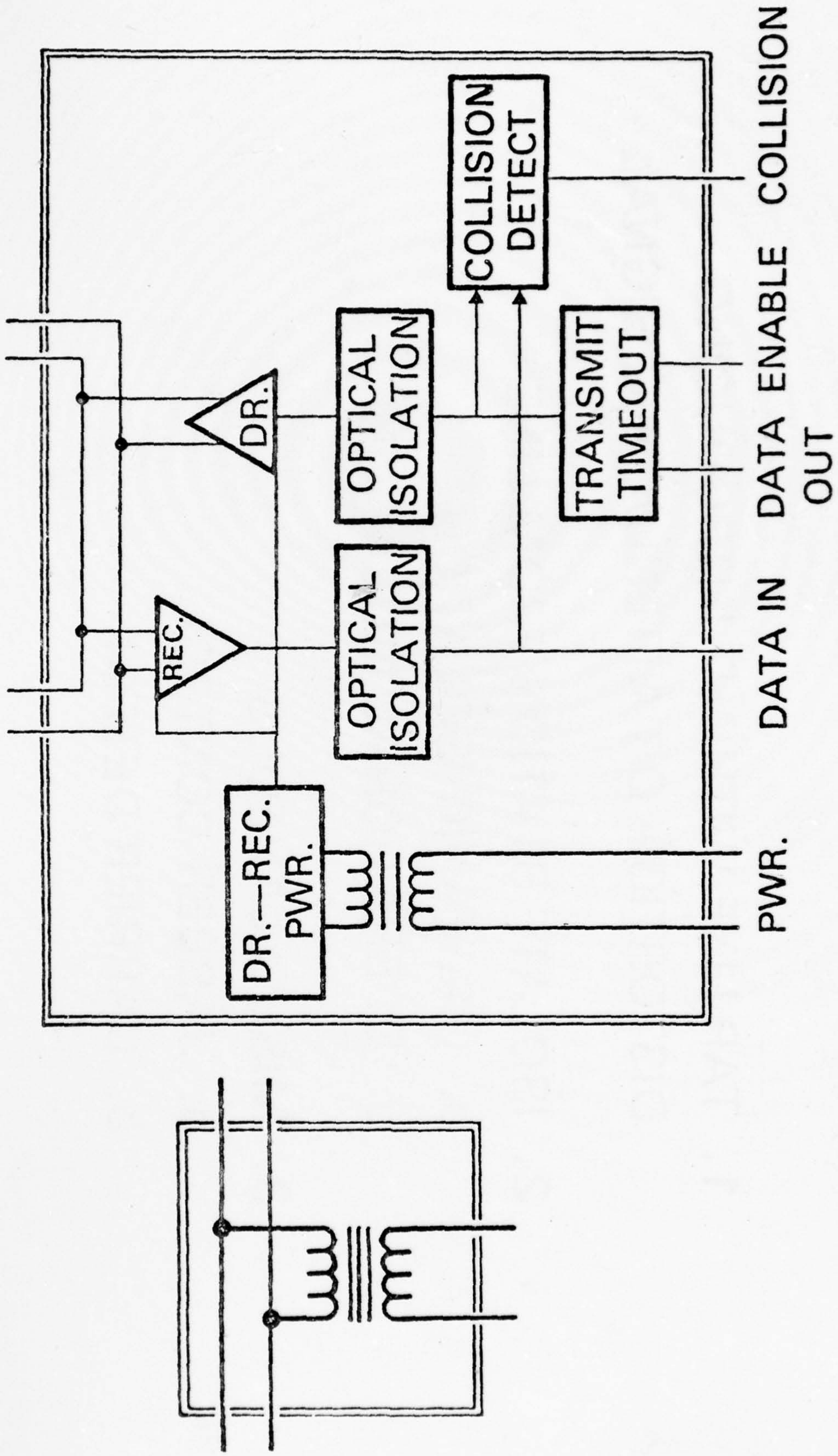

CAPACITANCE
10,000 μF
0.03 μF
UNSAFE
SAFE
VOLTS
10
100

CABLE DISTRIBUTION



Tap & Coupler Requirements

1. TAP LINE WITHOUT INTRODUCING DISTORTION OR ATTENUATING SIGNAL
2. ISOLATE STATION FROM LINE
   a. GALVANIC ISOLATION
   b. OPERATIONAL ISOLATION
3. MISCELLANEOUS
   a. JABBER CONTROL
   b. CARRIER DETECT



COLLISION DETECT
OPTICAL ISOLATION
OPTICAL ISOLATION
TRANSMIT TIMEOUT
DR.
REC.
DR.—REC. PWR.
PWR.
DATA IN
DATA OUT
ENABLE
COLLISION

NOISE IMMUNITY?

—Characteristics Of Industrial Noise?

—System Test Stimulus Methods?

—Desired System Characteristics?

—Measurement Methods?

—EMI-RFI Generated By Equipment?

—Are MIL-STD's Applicable?

	DATA RATE MBITS/SEC	DISTANCE KM.	# OF STATIONS	CABLE TYPE	MODULATION	COUPLING TAP	INTRINSIC SAFETY	NOISE MARGIN
BELL TI	1.544	1.8	1	TW. PR.	MLB-BP	X FORMER		
DEC PDM	0.056	4.8	62	SHIELDED TW. PR.	BI ϕ—S			
ETHERNET	3.0	1.0+	256	CATV COAX	MANCH.—BP	CATV X FORMER		
TDC-2000	0.25	1.5	63	COAX	MLB	X FORMER		
MIT CHAOS	8.0	0.5+	256	CATV COAX	MANCH.—BP	COUPLER BOX OPTO—ISOL.		
MIL 1553B	1.0	0.15 0.3	31 1	SHIELDED TW. PR.	MANCHESTER BI-PHASE	X FORMER		MIL-STD-461 MIL-STD-462 MIL-E-6051
NBS—NET	1.0	1.6	>256	CATV COAX	MANCHESTER BI-PHASE	COUPLER BOX OPTO.—ISOL.		
CENTUM F-BUS	0.25	2.0	30	COAX	BASEBAND	X FORMER		

SUMMARY OF RAW BIT RATE REQUIREMENTS FOR EQUIVALENT PROWAY FUNCTIONS & THROUGHPUT USING HDLC & TDC PROTOCOLS

HDLC HIGHER ($\leq$150%) FOR SHORT, PREDEFINED TRANSACTIONS ($\leq$6 DATA BYTES)

HDLC LOWER ABOVE 6 DATA BYTES. APPROACHES 50% FOR LONG MESSAGES.

HDLC SUBSTANTIALLY HIGHER IF EACH STATION IS ROUTINELY POLLED INDIVIDUALLY.

≈ EQUAL FOR POLLING IF HDLC USES GROUP POLLING WITH INDIVIDUAL POLLS TO RESOLVE CONTENTION.

CURRENT HDLC COMMANDS ARE VERY INEFFICIENT FOR PRIMARY TRANSFER (≈2000% HIGHER BIT RATE).

A COMPATIBLE NEW HDLC COMMAND MAKES PRIMARY TRANSFER ACCEPTABLE (200% HIGHER BIT RATE).

Comparison of Elementary Proway Operation

HDLC (Half-Duplex, Normal Response Mode) vs TDC 2000

Operation	TDC Frames	TDC Total Bits	HDLC Frames	HDLC Total Bits	HDLC Bits / TDC Bits
1. Active Talker requests 2 data bytes from a Responder with immediate Acknowledge.	CR_{WD}(No Ack) Talker Responder	64	$I(2)_{I(2)}{}^{RR}$	176	2.75
2. Active Talker requests 2 data bytes from a Responder with delayed Acknowledge (Note 1)	CR_{WD}(No Ack)	64	$I(2)_{I(2)}$	120	1.88
3. Active Talker requests 20 data bytes from a Responder with immediate Acknowledge (Note 2)	$CR_{9BD,WD}$(No Ack)	352	$I(3)_{I(20)}{}^{RR}$	328	0.93
4. Active Talker requests 200 data bytes from a Responder with immediate Acknowledge (Note 2)	$10CR_{10(9BD,WD)}$(No Ack)	3520	$I(3)_{I(200)}{}^{RR}$	1768	0.50
5. Active Talker writes 2 data bytes to a Listener with immediate Acknowledge.	CW, WD_{WDE}	96	$I(4)_{RR}$	128	1.33
6. Active Talker writes 20 data bytes to a Listener with immediate Acknowledge.	$CW, 9BD, WD_{WD}$	384	$I(22)_{RR}$	272	0.71

Comparison of Elementary Proway Operation

HDLC (Half-Duplex, Normal Response Mode) vs TDC 2000

Operation	TDC Frames	TDC Total Bits	HDLC Frames	HDLC Total Bits	HDLC Bits / TDC Bits
7. Active Talker writes 100 Data bytes to a Listener with immediate Acknowledge.	$CW,49BD,WD_{WD}$	1664	$I(102)_{RR}$	912	0.55
8. Active Talker writes a 2000 byte program to a Listener (Note 3)	$CW,WD_{WDE}CW,999BD,WD_{WDE}$	32160	$SIM_{UA}I(2002)_{DM}$	16208	0.50
9. Active Talker writes 2 bytes to and requests 2 bytes from a Responder with immediate acknowledge.	$CW,WD_{WDE}CW_{WD}$(No Ack)	160	$I(6)_{I(2)}{}^{RR}$	208	1.30
10. Active Talker writes 2 data bytes to and requests 2 data bytes from a Responder with delayed acknowledge (Note 1)	$CW,WD_{WDE}CW_{WD}$(No Ack)	160	$I(6)_{I(2)}$	160	1.0
11. Active controller individually polls 1 Talker and receives 4 data bytes from it (Note 4)	$PL_{PR,PR}CU_{CW,BD,WD_{WDE}}$ Cntrlr Talker Listnr	256	$RR(P)orUP(I)_{I(4)}{}^{RR}$	176	0.69
12. Active controller individually polls with Ack 4 Talkers and receives no response (Note 4)	$PL_{PR,PR}$	96	$4RR(P)_{4RR}$	384	4.0

Comparison of Elementary Proway Operation

HDLC (Half-Duplex, Normal Response Mode) vs TDC 2000

Operation \	TDC Frames	TDC Total Bits	HDLC Frames	HDLC Total Bits	HDLC Bits / TDC Bits
13. Active controller individually polls with Ack 4 Talkers and receives 4 data bytes from 1 Talker (Note 4)	$PL_{PR,PR}CU_{CW,BD,WD}WD_{WDE}$	256	$3RR(P)_{3RR}RR(P)_{I(4)}RR$	460	1.80
14. Active controller individually polls without Ack 16 Talkers and receives no response (Note 4, Note 5)	$PL_{PR,PR}$	96	16UP(I),16 Time Out	1152	12.0
15. Active controller individually polls without Ack 16 Talkers and receives 4 data bytes from 1 Talker (Note 4, Note 5)	$PL_{PR,PR}CU_{CW,DB,WD}WD_{WDE}$	256	$15UP(I),15\ Time\ Out,UP(I)_{I(4)}RR$	1256	4.91
16. Active controller individually polls without Ack 16 Talkers and receives 4 responses of 4 bytes each (Note 4, Note 5)	$PL_{PR,PR}4CU_{4(CW,DB,WD)}{}_{4WDE}$	736	$12UP(I),12\ Time\ Out,4UP(I)_{4I(4)}4RR$	1472	2.0

Comparison of Elementary Proway Operation

HDLC (Half-Duplex, Normal Response Mode) vs TDC 2000

	Operation	TDC Frames	TDC Total Bits	HDLC Frames	HDLC Total Bits	HDLC Bits / TDC Bits
17.	Active controller group polls all stations and receives no response (Note 4, Note 5, Note 6)	$PL_{PR,PR}$	96	UP(G), Time Out	72.	0.75
18.	Active controller group polls all stations and receives 4 data bytes from 1 Talker (Notes 4, 5, 6)	$PL_{PR,PR}{}^{CU}{}_{CW,DB,WD}{}_{WDE}$	256	$UP(G)_{I(4)}{}^{RR}$	180	0.70
19.	Active controller group polls 16 stations and receives 4 responses of 4 data bytes each. (Notes 4, 5, 6, 10)	$PL_{PR,PR}{}^{4CU}{}_{4(CW,DB,WD)}{}_{4WDE}$	736	$UP(G)_{Error(4)}{}^{12UP(I),12\ T.O.}\ 4UP(I)_{4I(4)}{}^{4RR}$	1600	2.17
20.	Transfer of Active Talker (& controller) status & system reinitialization with 20 stations (Notes 7, 8)	$Idle_{Active}$ Old Controller / New Controller	20	$UP(G)_{SNRM}{}^{UA}{}_{19SNRM}{}^{19UA}$	1968	98.5 (20.5 vs 96)
21.	Transfer of Active Talker status & system reinitialization with N stations & new HDLC compatible command. (Note 9)	$Idle_{Active}$	20	$UP(G)_{SNRM}{}^{UA}USIM(G)$	192	9.6 (2.0 vs 96)

Note 0 - In TDC the Highway Traffic Director is the Active Controller and any Preferred Access device can be the Active Talker. A polled device requires additional overhead as Active Talker since it is selected for only 1 transaction. In HDLC the Primary Station is both active Talker and Active Controller.

Note 0a - The HDLC system is based on Half duplex operation with Normal Response Mode only for secondaries.

Note 1 - Assumes that by convention HDLC primary stations do not send acknowledges immediately on receiving valid I frames. The acknowledge is included in the next frame sent to this station.

Note 2 - Assumes TDC stations operating with certain addresses predefined to request specific 20 byte data blocks.

Note 3 - This operation is not defined for TDC. A possible sequence is shown for comparison.

Note 4 - Polling under TDC has no security checks and is limited to 27 predefined stations.

Note 5 - Assumes HDLC polling with P bit = 0 so that no response is required from non-candidate Talkers. Time Out assumed = 24 bits. Similar to TDC in security checks.

Note 6 - HDLC group poll has same security as TDC poll but does not allocate a time slot for response. Time Out assume = 24 bits.

Note 7 - TDC stations require no initiatization on transfer of Active Talker status. Talker Transfer is accomplished by separate dedicated lines.

Note 8 - This function is not defined for HDLC by Intl standards. The sequence shown uses defined elements of procedure - DIS4335. A specific group poll address is predefined to allow transfer ot Talker status.

Note 9 - A new HDLC compatible command-Unnumbered Initialization - USIM with a Group Option is defined. This command resets the sequence numbers of all secondaries without changing their mode. It does not allow a response when the P bit = 0.

Note 10 - Assumes no time allocation mechanism on group polls so that multiple responses collide. Primary resolves conflicts with individual polls.

TUTORIAL NUMBER V

LONG RANGE FUNCTIONAL REQUIREMENTS AND DESIGN CRITERIA FOR OPERATOR CONSOLES

Mr. Remsi R. Messare
Exxon Research and Engineering Co.
Florham Park, New Jersey

Presented at the Purdue Americas Meeting

PRECEDING PAGE BLANK - NOT FILMED

LONG RANGE FUNCTIONAL REQUIREMENTS AND DESIGN CRITERIA FOR OPERATOR CONSOLES

TUTORIAL PRESENTATION ON
OPERATOR-TO-PROCESS INTERFACE

5TH AMERICAN REGIONAL MEETING
APRIL 1973
THE INTERNATIONAL PURDUE WORKSHOP
ON INDUSTRIAL COMPUTER SYSTEMS

BY
REMSI R. MESSARE
STAFF ENGINEER
TECHNOLOGY DEPARTMENT
EXXON RESEARCH & ENGINEERING CO.

NOTE

THIS PRESENTATION SUMMARIZES THE FINDINGS OF A STUDY ON OPERATOR INTERFACES CONDUCTED IN 1977 BY EXXON RESEARCH AND ENGINEERING COMPANY, EXXON CHEMICALS, USA AND IMPERIAL OIL ENTERPRISES LTD (IOEL)

FOR MORE DETAILS THE READER SHOULD REFER TO THE PAPER "TENTATIVE FUNCTIONAL REQUIREMENTS AND DESIGN CRITERIA FOR OPERATOR INTERFACES" BY R. R. MESSARE, J. H. LAROCHE AND D.SHANNON AND INCLUDED IN THE MINUTES OF THE 5TH ANNUAL MEETING OF THE INTERNATIONAL PURDUE WORKSHOP.

MAIN POINTS COVERED

- BRIEF SUMMARY OF:
  - SCOPE OF STUDY
  - MAIN SHORTCOMING AND INCENTIVES FOR IMPROVEMENT
- ANALYSIS OF KEY POINTS OF LONG RANGE REQUIREMENTS
  - SYSTEM CONFIGURATION
  - RELIABILITY ASPECTS
  - CONSOLE FUNCTIONALITY
    - + DISPLAY ORGANIZATION
    - + DISPLAY/KEYBOARD INTERACTION
    - + FLEXIBILITY
- SHORT RANGE OBJECTIVES
  - INTEGRATION BETWEEN COMPUTER CONSOLES AND INSTRUMENT CONSOLES

OPERATOR INTERFACES FOR PROCESS CONTROL

- REASONS FOR STUDY
  - SHORTCOMINGS WITH CURRENT OPERATOR INTERFACE DESIGN
  - IMPORTANT INCENTIVES FOR IMPROVEMENT
  - TECHNOLOGICAL EVOLUTION

- SCOPE
  - CONSOLIDATE AND DOCUMENT SHORTCOMING AND INCENTIVES IN THE AREA OF OPERATOR-TO-PROCESS INTERFACE
  - DEVELOP EXXON'S LONG RANGE FUNCTIONAL REQUIREMENTS FOR FUTURE OPERATOR CONSOLES (5-10 YEARS TIME FRAME)
  - DEVELOP STEPS TO MEET LONG RANGE GOALS

- DISSEMINATION OF LONG RANGE FUNCTIONAL REQUIREMENTS
  - ENHANCE UNDERSTANDING OF OUR GOALS
  - ESTABLISH AREAS OF COMMON INTEREST

HOW STUDY WAS CARRIED-OUT

- STUDY TEAM
- FAMILIARIZATION WITH COMMERCIAL CONSOLE DESIGNS/OTHER STUDIES
- PLANT SURVEYS (EXXON CIRCUIT)

MAIN SHORTCOMINGS WITH CURRENT OPERATOR INTERFACES

- FRAGMENTATION
  - NEED TO LEARN AND USE DIFFERENT INTERFACES/PROCEDURES TO ACCOMPLISH SAME FUNCTIONS
  - INEFFICIENT USE OF HARDWARE REDUNDANCY
  - DATA PRESENTATION OFTEN CONTRADICTORY AND/OR CONFUSING (ESPECIALLY PROCESS ALARMING)
- COMPLICATED AND SLOW CONSOLE PROCEDURES
  - DETER OPERATORS TO FULLY UTILIZE COMPUTER CAPABILITIES
  - COMPUTER CONSOLES USUALLY IMPRACTICAL FOR EMERGENCIES AND/OR START-UP

INCENTIVES FOR IMPROVEMENT

- IMPROVE OPERATOR EFFICIENCY
  - HIGHER UTILIZATION OF ADVANCED CONTROLS
  - IMPROVED OPERATOR RESPONSE TO ABNORMAL CONDITIONS
  - SIMPLIFY OPERATOR TRAINING REQUIREMENTS
- ELIMINATION OF UNNECESSARY HARDWARE REDUNDANCY
- IMPLEMENTATION OF DISPERSED INSTRUMENT TOPOLOGIES

LONG RANGE GOAL

BASED ON:

- ANALYSIS OF SHORTCOMINGS AND INCENTIVES
- FORECAST OF FUTURE NEEDS

AIMS AT:

- FULLY INTEGRATED CONSOLES (ELIMINATION OF BACK-UP INTERFACES)
  - CONSOLIDATION OF INTERFACES
  - UNIFICATION OF HARDWARE AND OPERATING PROCEDURES

FUNCTIONAL REQUIREMENTS:

- ADDRESS CONSOLES INTENDED AS THE SOLE OPERATOR INTERFACE
- EMPHASIS PLACED ON:
  - RELIABILITY ASPECTS
  - SIMPLE AND FAST CONSOLE PROCEDURES
  - HIERARCHICAL DISPLAY ORGANIZATION
  - FUNCTIONALITY AND DISPLAY CAPABILITY ESPECIALLY IN THE AREA OF ALARMING
- SEVERAL CONCEPTS CAN BE ADOPTED FOR CURRENT CONSOLE DESIGNS

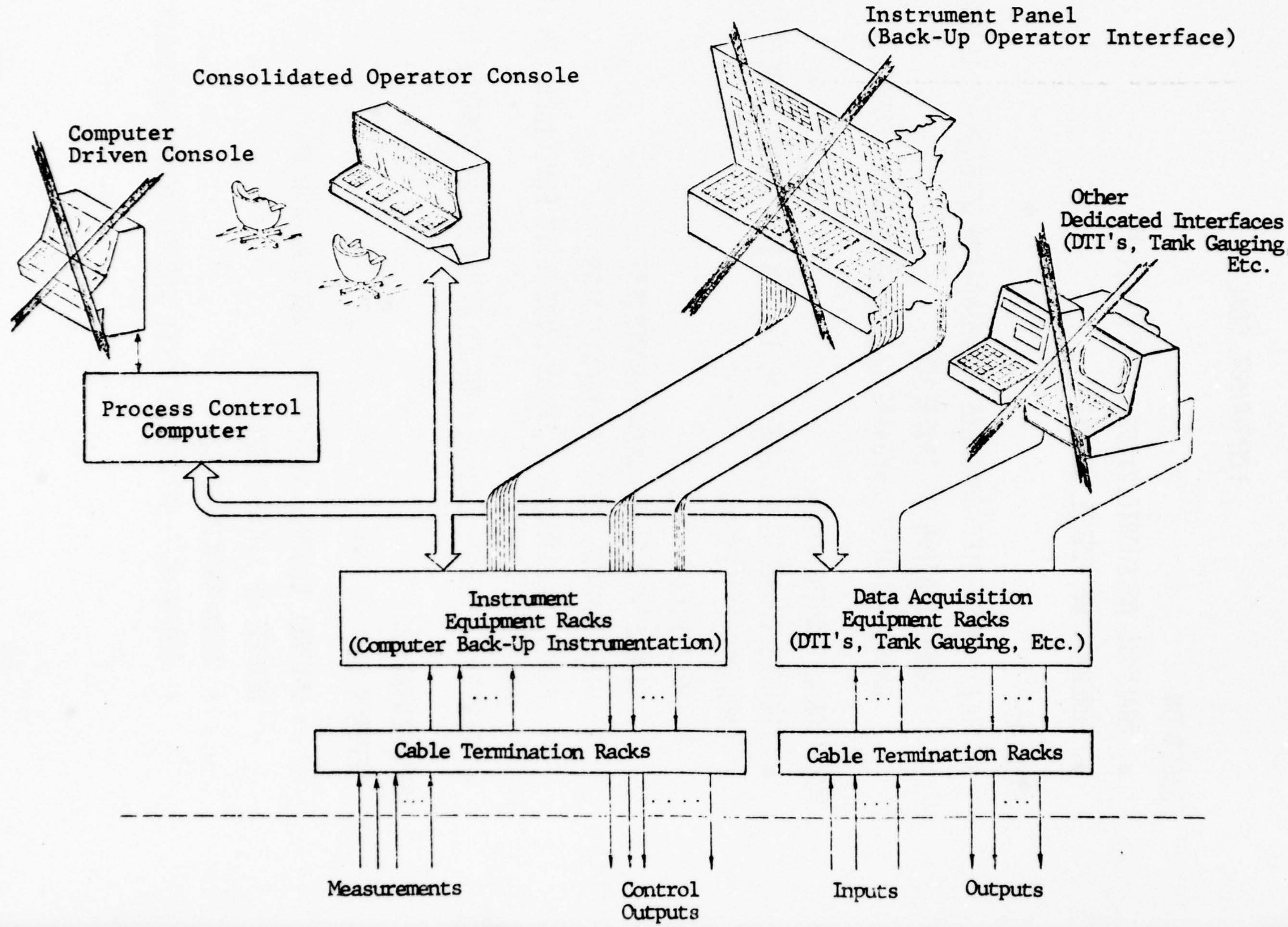
IMPLEMENTATION:

DEPENDS ON:

- OVERALL CONTROL SYSTEM ARCHITECTURE AND TOPOLOGY
- DEGREE OF STANDARDIZATION
  - + DATA TRANSMISSION
  - + INTERFACES BETWEEN DEVICES AND COMMUNCATION SUBSYSTEMS

FIGURE 1

CONSOLIDATION OF OPERATOR INTERFACES



CONSOLE AVAILABILITY

- SAME AS AVAILABILITY OF CURRENT INTERFACE (I. E. COMPUTER CONSOLE + BACK-UP INTERFACE)
- DESIGN CRITERIA
  - EXCLUDE (STATISTICALLY) TOTAL FAILURE OF THE OPERATOR INTERFACE
  - DISPLAY AND INTERACTIVE FEACTURES TO REMAIN INTACT UNDER PARTIAL FAILURES WITH MINIMUM DEGREE OF DEGRADATION
  - ABILITY OF CONSOLE TO COMMUNICATE ON ITS OWN WITH ALL THE COMPONENTS OF THE CONTROL SYSTEM

CONSOLE MODULARITY

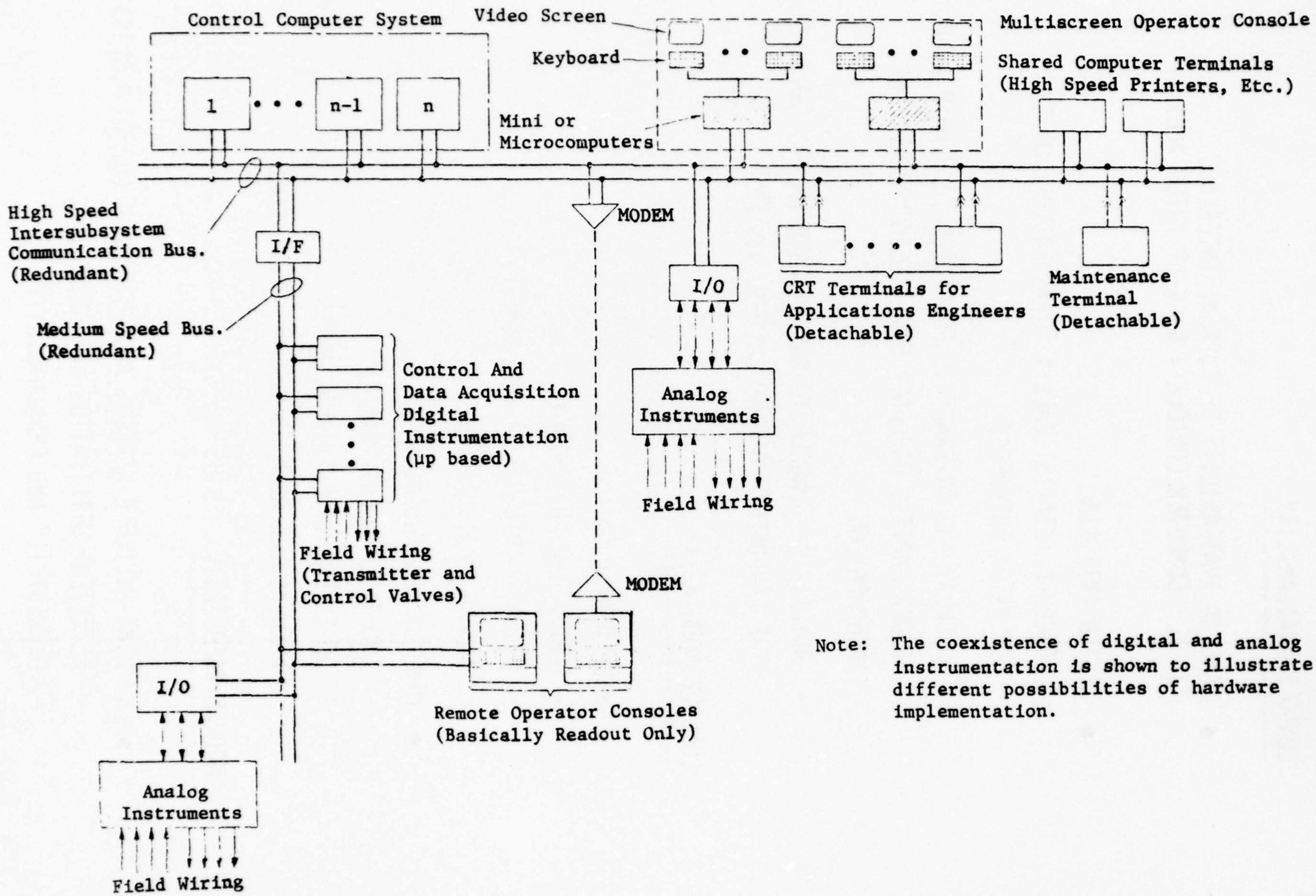
- WILL DEPEND ON
  - OPERATING PHILOSOPHY
  - PROCESS INTEGRATION AND COMPLEXITY
  - CONSOLE FAILURE MODES
- MINIMUM CONFIGURATION
  - TWO INDEPENDENT MODULES
  - NO INTERACTIVE FAILURE MODES
  - EACH MODULE ABLE TO SUPPORT TOTAL LOAD

MODULE CAPABILITY

- SUPPORT MULTIPLE SCREENS AND KEYBOARDS (UP TO 4 SETS) FOR:
  - EFFICIENT UTILIZATION OF HARDWARE
  - FLEXIBILITY AND EXPANDIBILITY

FIGURE 2

SCHEMATIC CONFIGURATION OF DISTRIBUTED/HIERARCHICAL CONTROL SYSTEM



Note: The coexistence of digital and analog instrumentation is shown to illustrate different possibilities of hardware implementation.

CONSOLE HARDWARE CONFIGURATION

- 4 - 6 VIDEO SCREENS/KEYBOARDS
- 2 - 3 MODULES
- INCLUDE "HARDWIRED" COMPONENTS (AS NEEDED)
  - DEDICATED ALARM ANNUNCIATORS
  - DEDICATED PUSHBUTTONS, ETC.

OPERATOR WORKLOAD AND CONSOLE CAPACITY

- CURRENT WORKLOAD/CONSOLE OPERATOR:
  - VARIES CONSIDERABLY (70-250 CONTROL OUTPUTS)
  - DEPENDS ON NATURE AND COMPLEXITY OF PROCESS
- EXPECTED FUTURE WORKLOAD
  - BASICALLY SAME AS TODAY
  - 300 CONTROL OUTPUTS MAX.
- CONSOLE DISPLAY CAPACITY DEPENDS ON:
  - EXPECTED LOAD OF FUTURE COMPUTER CONTROL SYSTEMS
  - PLANT SIZE
  - OPERATING PHILOSOPHY

LOAD ANALYSIS FOR FUTURE COMPUTER SYSTEMS

TYPE OF TAGS	PLANT SIZE: SMALL	MEDIUM	LARGE	X-LARGE
Control and Non-Control Inputs	Approx. 11% of X-Large	Approx. 27% of X-Large	Approx. 50% of X-Large	2800-3000
Additional Miscellaneous Inputs (TC Inputs, Tank Levels, etc.)				3000-3200
On-Off Alarm Inputs				1600-2000
Other On-Off Variables (MOV's, Pumps, Mixers, etc.)				400-600
Calculated Variables (Control & Non-Control)				1500-1800
Lab Inputs				350-400
TOTAL (See Note 3)	≃ 1150	≃ 2800	≃ 5300	≃ 10400
Approx. Number of Control Loops	110	250	450	950

NOTES:
1. The above numbers represent estimated averages and should be treated as order of magnitudes for sizing purposes.
2. The majority of cases fall between "medium" and "large"
3. Totals are approximately 11 times the corresponding number of control "loops" (i.e., control points with valve output).

CONSOLE LOADING CONDITIONS FOR TYPICAL LARGE SIZE INSTALLATION

TYPE OF TAGS		
Control Inputs with Valve Output (High Level 4-20 m A)	500	
Non Control Inputs (4-20 m A)	900	(Note 1)
DTI Inputs (Thermocouples)	1500	
Alarm Points	900	(Note 2)
Calculated Control Points (PID's or Cascade Primaries)	200	
Other Calculated Variables	780	(Note 3)
Lab Inputs	200	
Other Miscellaneous Inputs	400	
TOTAL	5380	

NOTES:

1. This figure is nearly twice as high than it is in current installations and takes into account the expected increase of non control type inputs in future computer systems. Today the sum of both control and non control type inputs (4-20 m A) would not exceed 900.

2. This figure represents the number of alarm points that in current installations are being displayed on panel mounted annunciators. Many of them are generated by the back-up instruments and not by field mounted sensors.

3. This figure represents approx. 60% of the total number of calculated variables. It is estimated that the rest may not be of interest to the process operator but rather to the applications engineer.

CONSOLE FUNCTIONALITY

- SHOULD ADDRESS NEEDS OF PROCESS OPERATORS
- SHOULD NOT BE PENALIZED TO SATISFY NEEDS OF OTHER WORKING GROUPS
  - MAINTENANCE
  - CONTROL APPLICATIONS
  - ETC.
- LOGS AND REPORTS
  - MAINLY GENERATED, STORED AND PRESENTED ELSEWHERE
  - MINIMUM NUMBER DISPLAYED ON CONSOLE (ASSOCIATED WITH OPERATOR DAILY TASKS)
- HARD COPY OF VIDEO DISPLAYS
- EMPHASIS ON PROCESS ORIENTED DISPLAYS
  - OVERVIEW CAPABILITY
    - + PLANT LEVEL (250-300 TAGS/DISPLAY)
    - + UNIT LEVEL (50-60 TAGS/DISPLAY)
    - + IMPORTANT ALARM CONDITIONS
      - PRIORITY
      - STATUS
  - HIERARCHICAL DISPLAY ORGANIZATION
  - MULTIPLE CHOICE FOR ALARM PRESENTATION AND ACKNOWLEDGEMENT
  - FAST AND SIMPLE CONSOLE PROCEDURES
    - + SINGLE ACTION
    - + FAST RESPONSE

ORGANIZATION OF BASIC DISPLAYS

BASIC DISPLAYS

- THREE MAIN HIERARCHICAL DISPLAY LEVELS
  - PLANT LEVEL
  - UNIT LEVEL
  - "LOOP" LEVEL (INTERACTIVE)
- ALARM DISPLAYS
- TREND DISPLAYS

ORGANIZATION

- BUILT AROUND THE CONCEPT OF "GROUP"
- 8 (OR 16) TAGS/GROUP (PREFERABLY 8)

PASSAGE/ACCESS

- PRIMARY PATHS } VIA SINGLE ACTION
- SECONDARY PATHS } VIA SINGLE ACTION
- BIDIRECTIONAL "PAGING"

PRIMARY PATHS

- "PLANT LEVEL" OVERVIEW AND "LOOP LEVEL"
- "UNIT LEVEL" OVERVIEW AND "LOOP LEVEL"
- ALARM DISPLAYS AND "LOOP LEVEL"

SECONDARY PATHS (VIA DISPLAY ASSOCIATION)

- "UNIT LEVEL" OVERVIEW AND "LOOP LEVEL"
- "UNIT LEVEL" OVERVIEW AND TREND DISPLAYS
- "LOOP LEVEL" AND TREND DISPLAYS
- BETWEEN TREND DISPLAYS
- BETWEEN MULTITAG (GROUP) DISPLAYS ("LOOP LEVEL")

PAGING

- OVERVIEW DISPLAYS (PLANT AND UNIT LEVEL)
- MULTITREND DISPLAYS
- MULTITAG DISPLAYS

FIGURE 3

HIERARCHICAL ORGANIZATION OF BASIC DISPLAYS

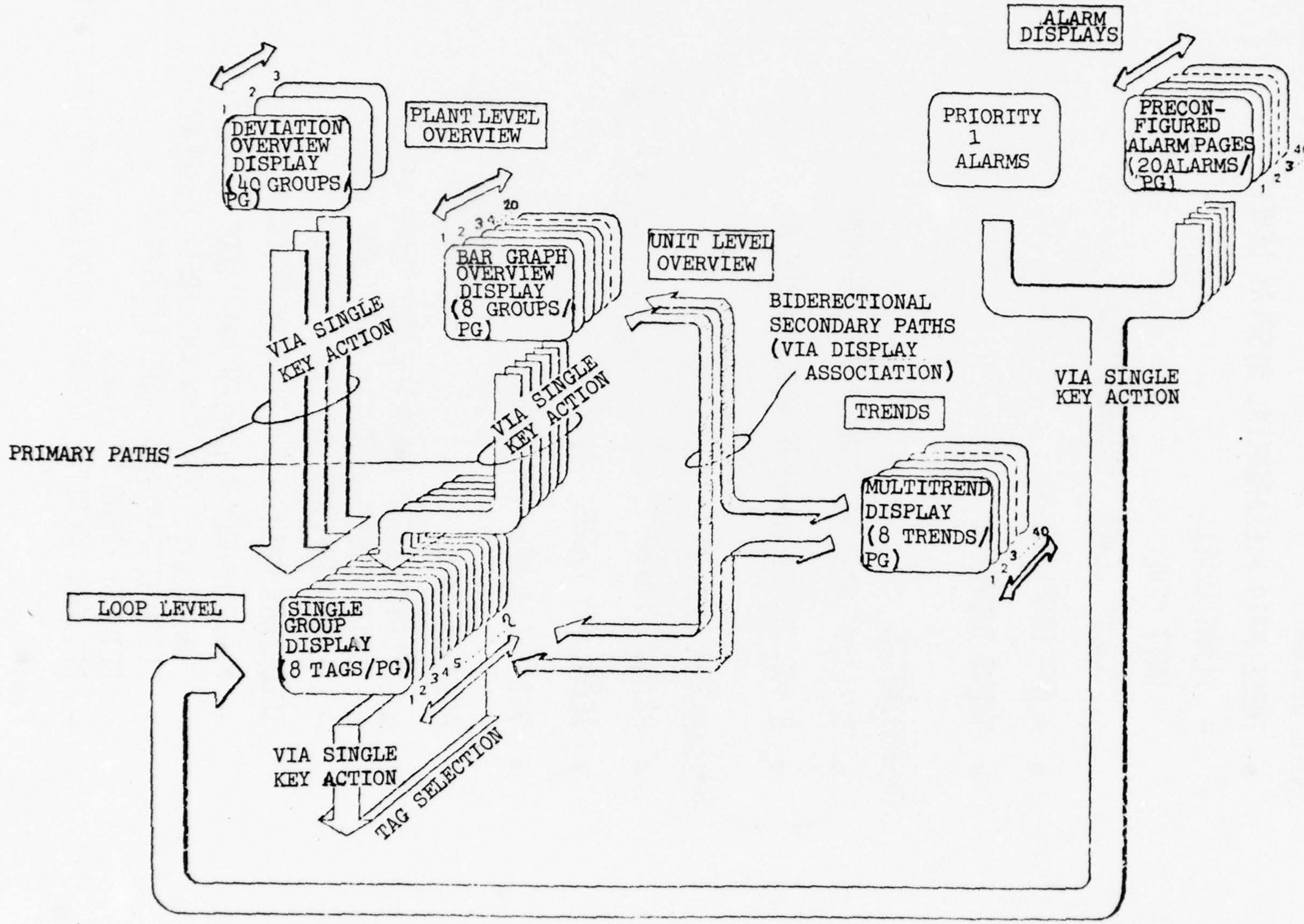
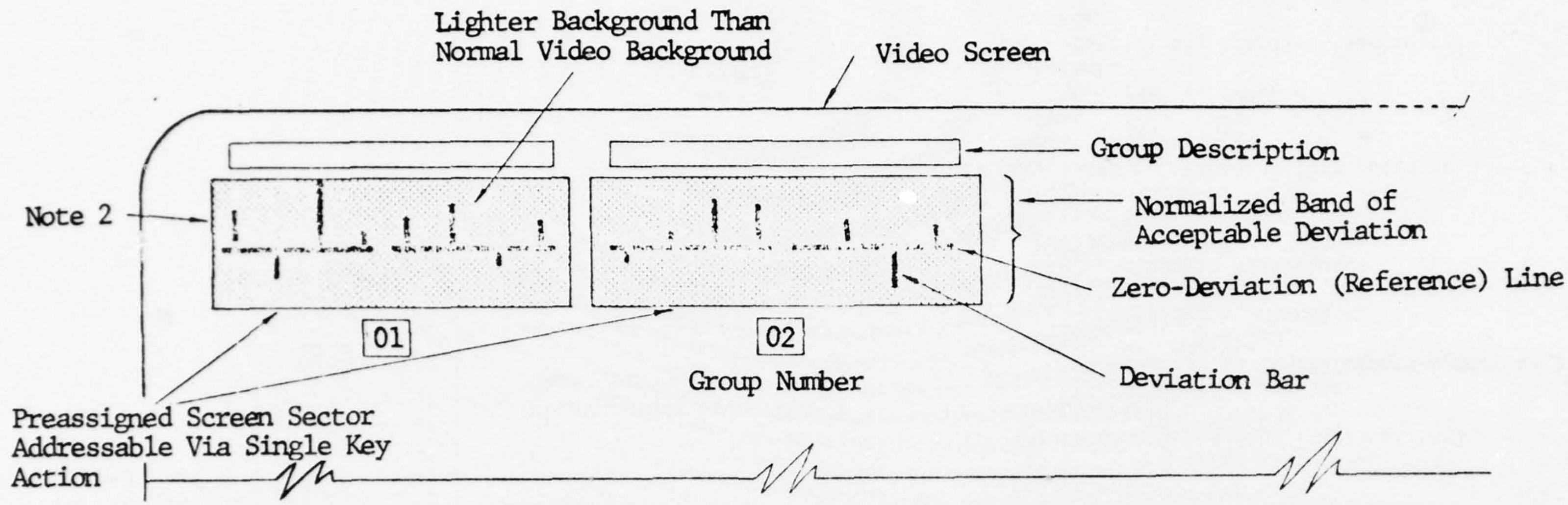


FIGURE 4

EXAMPLE OF PLANT LEVEL OVERVIEW DISPLAY



Notes:

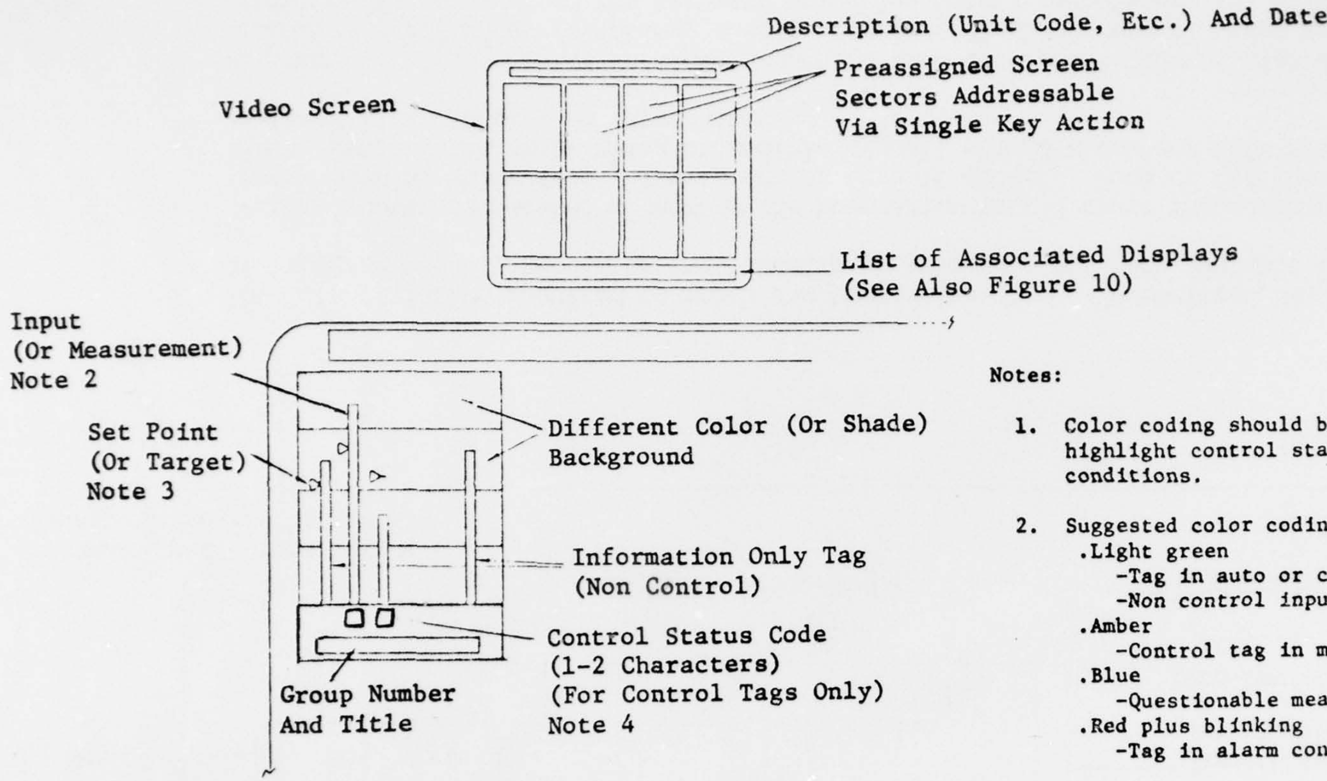
1. The % in deviation required to reach the boundaries of the "Normalized Band" (shown preferably in "reverse video") should be individually programmable for each variable form 0% to 100%.

2. Vertical bars (deviation) exceeding the boundaries should cause the corresponding group to change display color (e.g., from green or cyan to amber). None of the characteristics of alarm status (such as blinking or audible signal) should accompany this warning unless the exceeded limit is also an alarm limit.

3. A group containing tag(s) in alarm condition (low or high PV, low or high deviation, etc.) should be highlighted (blinking, change of color, flashing reverse video zone, etc.). The corresponding element of the keyboard selection matrix (see Figure 10) will initiate an ISA-1 alarm flashing sequence.

FIGURE 5

EXAMPLE OF UNIT LEVEL OVERVIEW DISPLAY



Notes:

1. Color coding should be used to further highlight control status and alarm conditions.

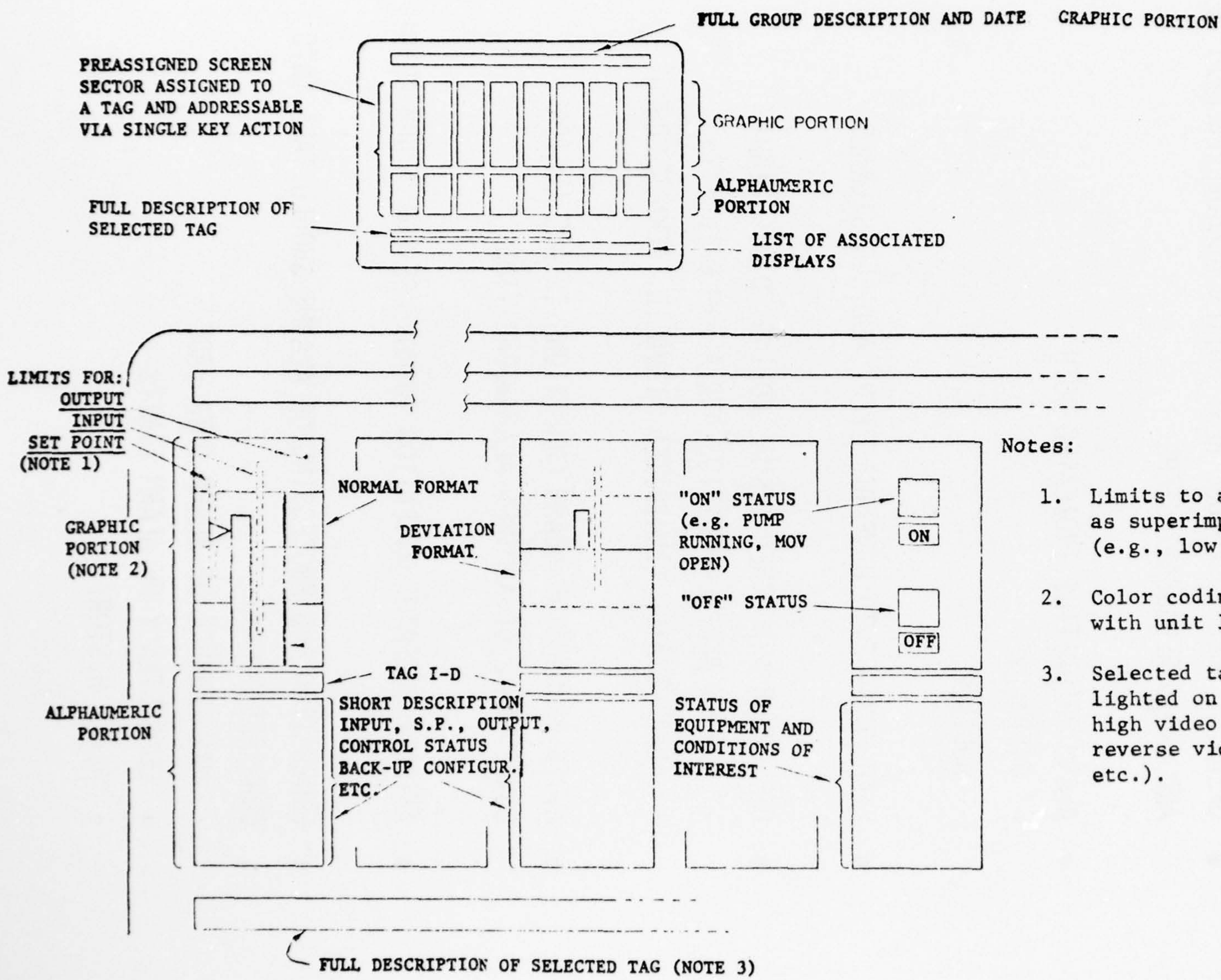
2. Suggested color coding (vertical bar)
   .Light green
      -Tag in auto or cascade
      -Non control input (Information Only tag)
   .Amber
      -Control tag in manual
   .Blue
      -Questionable measurement
   .Red plus blinking
      -Tag in alarm condition

3. Suggested color coding (set point or target pointer)
   .Cyan
      -Control tag in auto
   .Magenta
      -Control tag cascaded (secondary)
   .Amber
      -Manual target (non control point)

4. Symbolic and color coding should be used to indicate:
   .Tag off scan (by computer system)
    Suggested color: blue
   .Important cascade opened (for selected tags only)
    Suggested color: red

FIGURE 6

EXAMPLE OF MULTITAG FORMAT



Notes:

1. Limits to appear only on demand as superimposed information (e.g., low video intensification).

2. Color coding to be consistent with unit level overview.

3. Selected tag to be clearly highlighted on the screen (e.g., high video intensification, reverse video, symbolic coding, etc.).

MAIN REQUIREMENTS FOR CONSOLE ALARMING

- MULTIPLE CHOICES TO PRESENT AND ACKNOWLEDGE ALARM CONDITIONS
- OPERATOR FREE TO CHOOSE BEST SUITED ACKNOWLEDGEMENT METHOD AND AVOID REDUNDANT STEPS
- ALARM PRIORITIES INDICATED BOTH
  - VISUALLY
  - AUDIBLY
- EACH CONSOLE TO HAVE DIFFERENT AUDIBLE SIGNALS
- USE COLOR AND OTHER VISUAL CODING TO HIGHLIGHT:
  - TAG IN ALARM CONDITION AND ALARM PRIORITY
  - ALARM STATUS (INHIBITED, INVALID, ACKNOWLEDGED)
- AVOID PRESENTING ALARM CONDITIONS IF ARE LOGICAL CONSEQUENCE OF ANOTHER ALARM CONDITION
- ATTRACT OPERATOR ATTENTION TO MORE SEVERE ALARM CONDITION
- GENERATION AND PROCESSING OF ALARMS SHOULD TAKE INTO CONSIDERATION:
  - MAINTENANCE OF ALARM DATA BASE
  - CONSISTENCY OF ALARM LIMITS
  - ALARM HISTORY

CONSOLE ALARMING

- TWO DISPLAY PRIORITIES
  - PRIORITY 1: CRITICAL ALARMS

    IMMEDIATE IDENTIFICATION OF CONDITIONS THAT MAY RESULT IN:
    + INJURY TO PERSONNEL
    + DAMAGE TO PROCESS EQUIPMENT
    + LARGE OPERATING DEBITS
  - PRIORITY 2: IMPORTANT ALARMS

    ASSIST OPERATOR IN INITIATING ACTIONS TO PREVENT DEVELOPMENT OF SERIOUS UPSETS

- PRIORITIES BASED ON:
  - IMPORTANCE
  - RESPONSE TIME

- ALARM MESSAGES
  - ALL OTHER ABNORMAL CONDITIONS
  - VARIABLE FORMATS TO SUIT SPECIFIC CASES
    + VISUAL WARNINGS
    + TEXT MESSAGES

ALARM PRESENTATION

- "DEDICATED" ALARM DISPLAYS BUT ALSO ON OTHER DISPLAY FORMATS

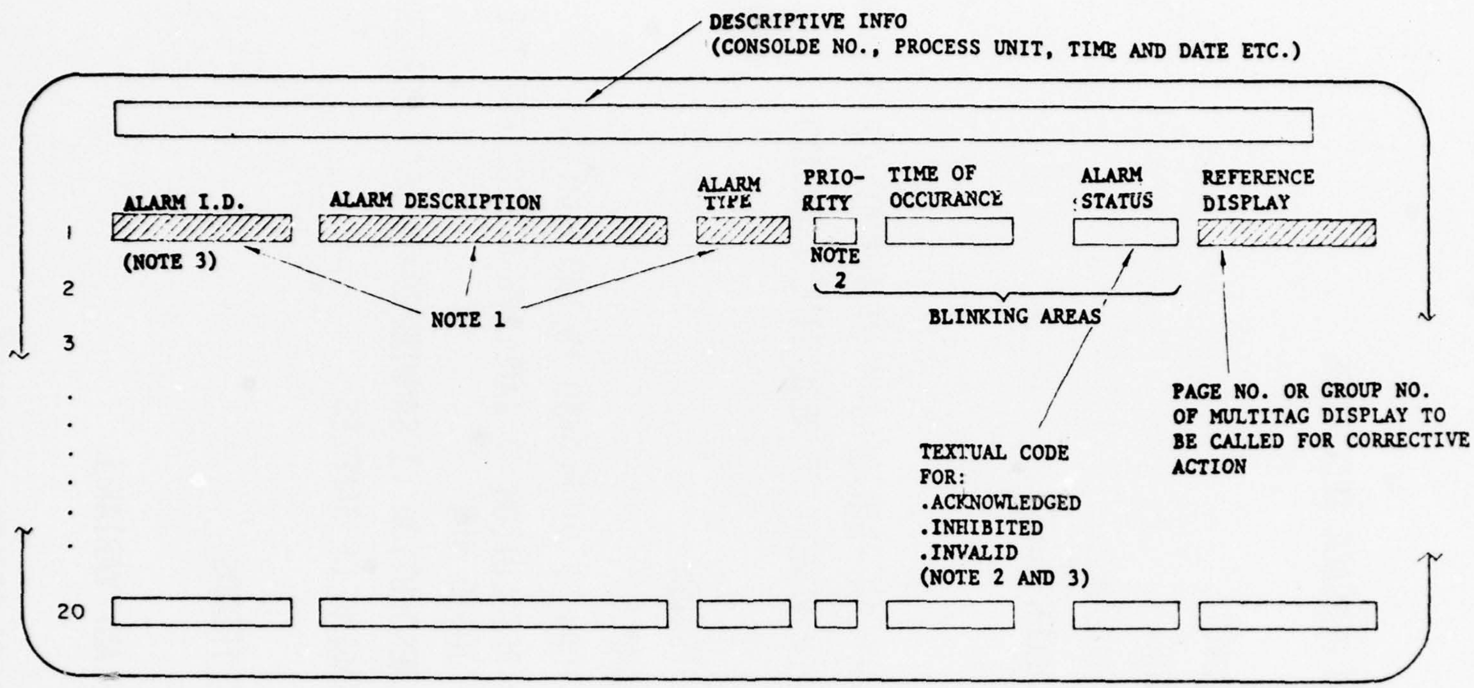
- TWO "DEDICATED" ALARM DISPLAYS
  - SAME FORMAT AT INDIVIDUAL ALARM LEVEL FOR BOTH DISPLAYS
  - DIFFERENT METHOD OF PRESENTATION AND INTERACTION WITH CONSOLE KEYBOARD

- FIRST DISPLAY
  - USED ONLY FOR PRIORITY 1 ALARMS
  - ALARMS APPEAR AUTOMATICALLY (NO ACTION BY OPERATOR)

- SECOND DISPLAY
  - USED FOR BOTH PRIORITIES
  - EMULATES DISPLAY PHILOSOPHY OF ALARM ANNUNCIATORS
  - PREARRANGED PAGES
  - OPERATES IN CONJUNCTION WITH A MULTIPURPOSE KEYBOARD MATRIX (ACTS AS "GROUP" ALARM)

FIGURE 7

EXAMPLE OF ALARM VIDEO PAGE



NOTES:

1. Blinking should be avoided for information displayed in shaded areas. Entire text however should be highlighted (e.g., via video intensification; reverse video technique, etc.)

2. Color coding should be used to further highlight alarm priority and alarm status.

3. Alarm I.D. and alarm status should remain intensified for acknowledged, inhibited and invalid alarms.

4. All alarms should be displayed, whether in alarm condition or not.

TREND CAPABILITY

TREND FORMATS TO REPLACE RECORDERS

- DISPLAY FORMAT
  - MULTITREND
  - PRECONFIGURABLE
  - EASILY ACCESSIBLE
  - SIMPLE

- TRENDS TO APPEAR WITH 2 HOUR (MINIMUM) HISTORY AND AUTOMATICALLY REVERT TO REAL TIME TRENDING

- DISPLAYED HISTORY
  - 1 DOT/SAMPLE
  - HISTORY CAN BE DISPLAYED IN TWO PORTIONS
    + HIGH RESOLUTION (1 SAMPLE/5-6 SEC) FOR THE LAST 10-15 MIN
    + LOW RESOLUTION (1 SAMPLE/20-30 SEC) FOR THE REST
  - NO AVERAGING OF SAMPLES

GENERAL PURPOSE TRENDS

- HISTORY OF ANY VARIABLE

- VARIABLE TIME SCALES AND RANGE

GRAPHIC CAPABILITY

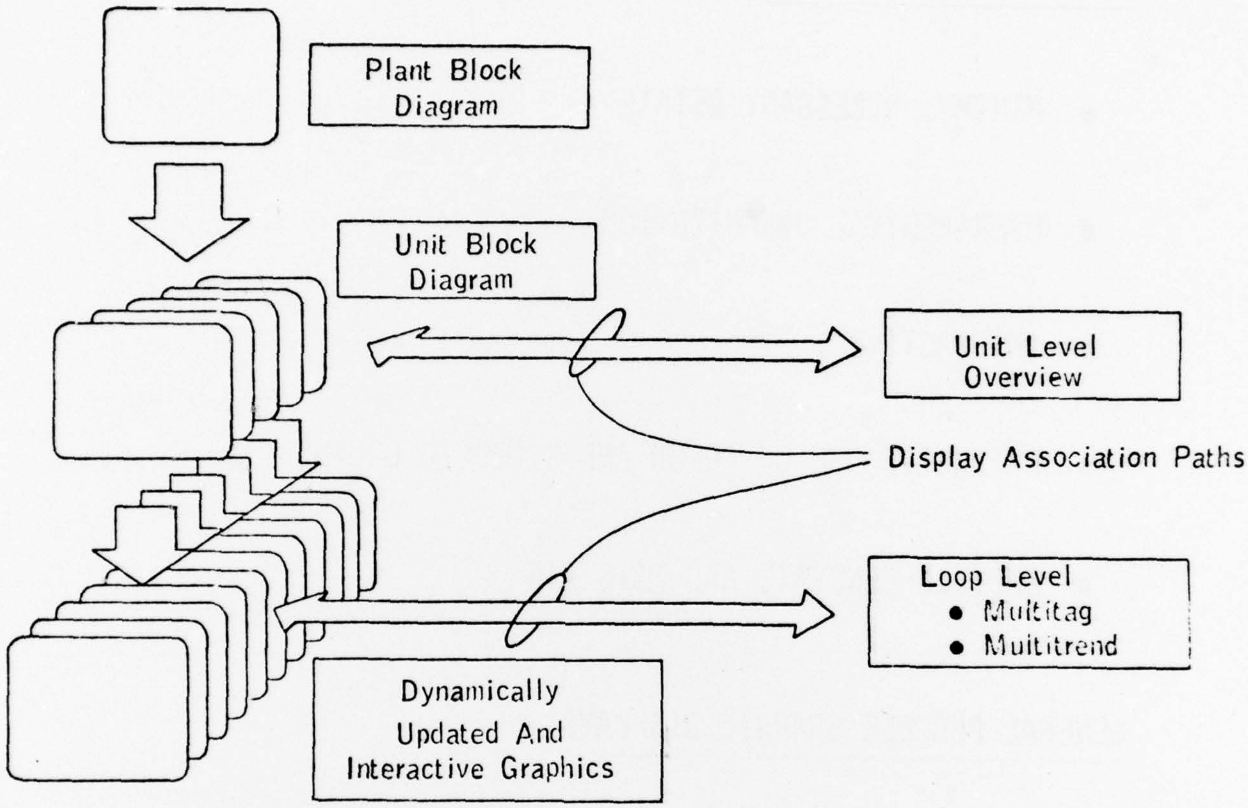
PROCESS FLOW DIAGRAMS

- MINIMUM NECESSARY DETAIL
- HIERARCHICAL ORGANIZATION
- INTERACTIVE
- EXTENSIVE USE OF COLOR AND SYMBOLIC CODING
- EASY TO GENERATE AND MAINTAIN

GENERAL PURPOSE GRAPHIC DISPLAYS

- BLOCK DIAGRAMS
  - CONTROL SCHEMES
  - SEQUENCE CONTROL
- EXPERIMENTAL DISPLAYS
  - USER DEFINED

FIGURE 8
HIERARCHICAL ORGANIZATION OF GRAPHIC DISPLAYS

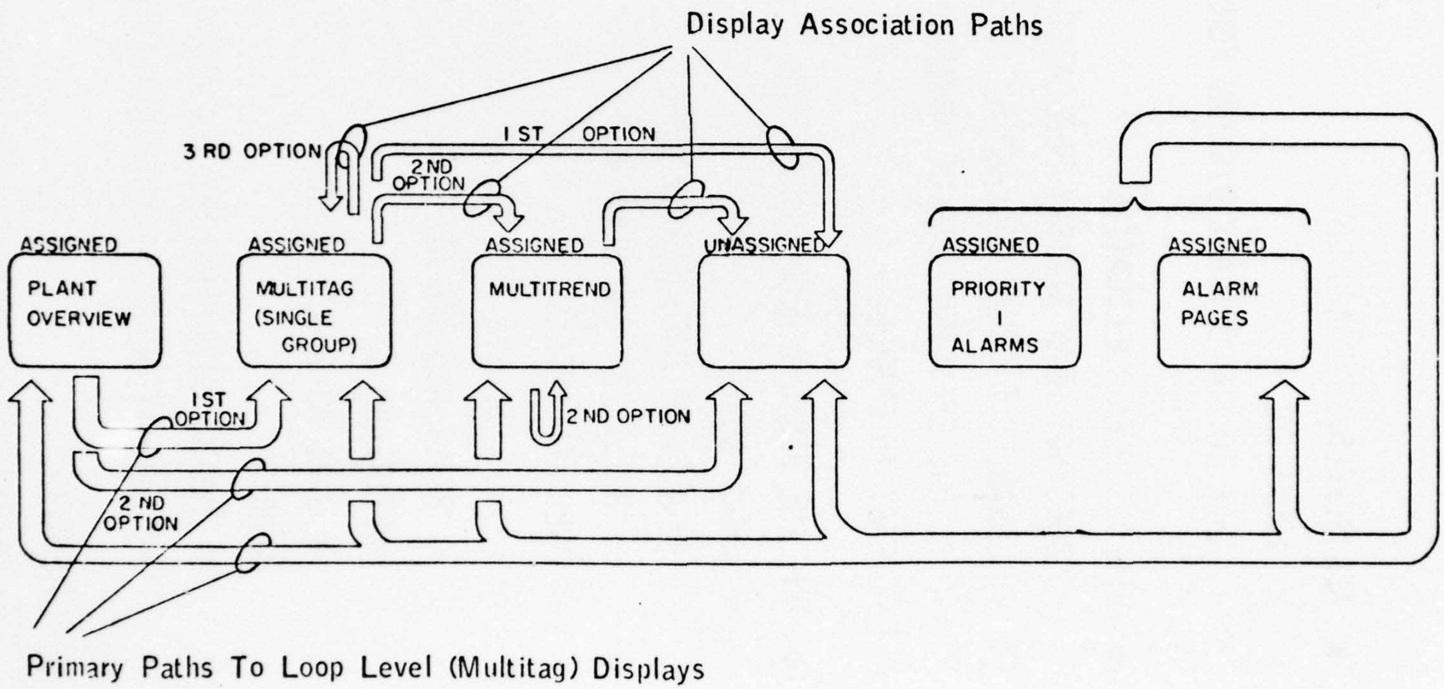


OVERALL CONSOLE FLEXIBILITY

- ALL SCREEN/KEYBOARD SETS
  - IDENTICAL
  - ABLE TO SUPPORT ANY DISPLAY FORMAT
- VIDEO SCREEN "ALLOCATION"
  - DOES NOT MEAN RIGIDLY DEDICATED SCREENS
  - ENHANCES DISPLAY CALL-UP BY "DISPLAY ASSOCIATION"
  - ALLOWS EFFICIENT USE OF "PRIMARY" PATHS
  - EASY TO CHANGE
- ON-LINE DISPLAY RECONFIGURATION
  - GROUP LEVEL
    + ADD, DELETE, CHANGE TAGS
    + BUILT AND ADD NEW GROUPS
  - OVERVIEW LEVELS
    + ADD, DELETE AND MODIFY GROUP ARRANGEMENT
    + BUILT AND ADD NEW PAGES
  - MULTITREND
    + SAME AS GROUP LEVEL
  - ALARMS
    + MODIFY ALARM PRIORITIES AND LIMITS
    + INHIBIT ALARMS (INDIVIDUALLY OR IN GROUPS)
- INTERLOCKS
  - DECISION FOR COMMISSIONING SHOULD BE LEFT TO THE USER

FIGURE 9

EXAMPLE OF SCREEN ALLOCATION



DISPLAY ADDRESSING

- FAST (SINGLE ACTION)
- FLEXIBLE (MULTIPURPOSE)

(RE: FIGURE 3 AND 10)

SELECTION KEYS

- TAG SELECTION (ON GROUP DISPLAY)
- GROUP SELECTION/CALL-UP (PRIMARY PATH FROM UNIT LEVEL OVERVIEW)

KEYBOARD MATRIX

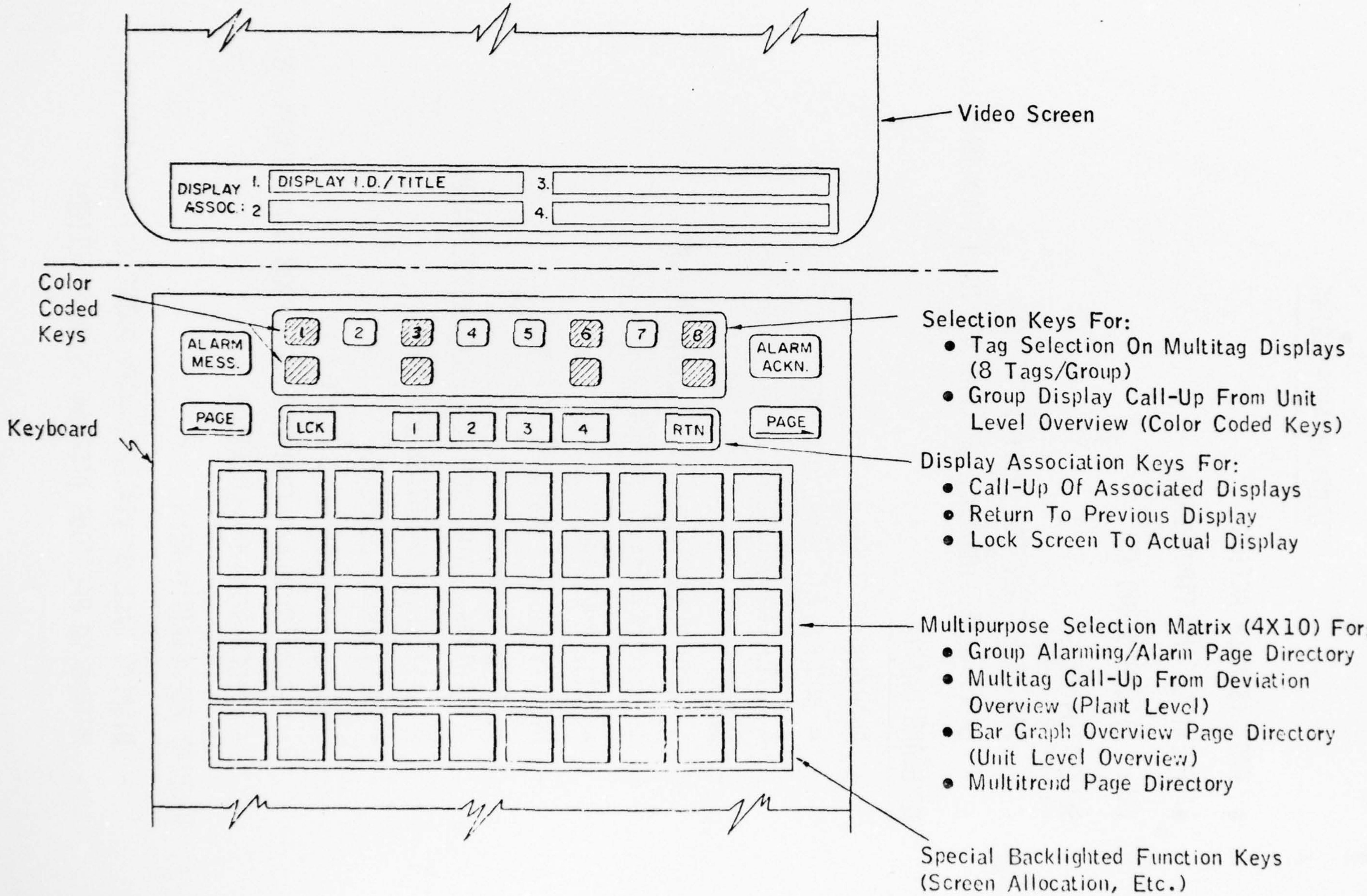
- ALARMING FEATURES
  - + WITH PLANT LEVEL OVERVIEW
  - + WITH PRECONFIGURED ALARM PAGES
- DIRECTORY
  - + WITH PLANT LEVEL OVERVIEW DISPLAYS
  - + WITH UNIT LEVEL OVERVIEW DISPLAYS
  - + WITH MULTITREND DISPLAYS
- DISPLAY CALL-UP (SINGLE ACTION)
  - + GROUP DISPLAYS (MULTITAG)
    (PRIMARY PATH FROM PLANT LEVEL OVERVIEW)
  - + UNIT LEVEL OVERVIEW DISPLAYS
  - + MULTITREND DISPLAYS
- SPECIAL FUNCTION KEYS (LAST ROW)
  - + GROUP DISPLAYS (PRIMARY PATH FROM ALARM DISPLAYS)
  - + VIDEO SCREEN ALLOCATION

DISPLAY ASSOCIATION KEYS

- DISPLAY CALL-UP VIA SECONDARY PATHS (SINGLE ACTION)
- RETURN TO PREVIOUS DISPLAY (SINGLE ACTION)

FIGURE 10

EXAMPLE OF SINGLE ACTION ADDRESSING KEYS



COLOR AND OTHER VISUAL CODING

- COLOR SHOULD BE PRIMARILY TO CONVEY INFORMATION
- COMBINATION OF COLOR AND OTHER VISUAL CODING MOST EFFECTIVE WAY
- ENVIRONMENTAL CONDITIONING OF HUMANS TO BE KEPT IN CONSIDERATIONS
- COLOR CODING SHOULD NOT BE USED ALONE TO CONVEY CRITICAL INFORMATION
- BLINKING SHOULD BE LIMITED TO ALARM CONDITIONS

SHORT RANGE OBJECTIVES

INTEGRATION BETWEEN COMPUTER AND INSTRUMENT VIDEO CONSOLES

- UNIFY HARDWARE DESIGN OF VIDEO SCREENS AND KEYBOARDS
- UNIFY CONSOLE PROCEDURES
- SWITCHOVER OF INSTRUMENT VIDEO CONSOLES BETWEEN MAIN COMPUTER AND OWN μP.
- NEED FOR STANDARDIZATION OF DATA TRANSMISSION SYSTEMS AND INTER SYSTEM INTERFACING TO ACHIEVE INTEGRATION